NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

MONITORING THE CALIBRATION OF A TORPEDO TEST RANGE

by

Sukru Korlu

September 1988

Thesis Advisor: R. R. Read

REPORT DOCUMENTATION PAGE

1a Report Security Classification Unclassified	1b Restrictive Markings
2a Security Classification Authority	3 Distribution Availability of Report Approved for public release; distribution is unlimited.
2b Declassification Downgrading Schedule	
4 Performing Organization Report Number(s)	5 Monitoring Organization Report Number(s)
6a Name of Performing Organization Naval Postgraduate School — 6b Office Symbol *(if applicable)* 55	7a Name of Monitoring Organization Naval Postgraduate School
6c Address *(city, state, and ZIP code)* Monterey, CA 93943-5000	7b Address *(city, state, and ZIP code)* Monterey, CA 93943-5000
8a Name of Funding Sponsoring Organization — 8b Office Symbol *(if applicable)*	9 Procurement Instrument Identification Number
8c Address *(city, state, and ZIP code)*	10 Source of Funding Numbers Program Element No — Project No — Task No — Work Unit Accession No

11 Title *(include security classification)* MONITORING THE CALIBRATION OF A TORPEDO TEST RANGE

12 Personal Author(s) Sukru Korlu

13a Type of Report	13b Time Covered	14 Date of Report *(year, month, day)*	15 Page Count
Master's Thesis	From To	September 1988	77

16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 Cosati Codes			18 Subject Terms *(continue on reverse if necessary and identify by block number)*
Field	Group	Subgroup	least squares, multivariate regression, calibration, array, theses. [illegible]

19 Abstract *(continue on reverse if necessary and identify by block number)*

Post-test analysis of vehicle tracking data taken from a three dimensional underwater range requires the splicing together of several pieces of track individually produced by distinct transducer arrays. The locations and orientations of the arrays must be known precisely in order to convert locally determined tracks into a coherent record in the general range coordinate system. The maintenance of calibration of the system is a problem and this study presents some least squares methodology that uses the tracking data itself to monitor it. More specificially, methodologies are developed to improve upon the array displacement and orientation correction algorithms and to estimate timing synchronization offsets. Applications are made to real data.

20 Distribution Availability of Abstract [✓ unclassified unlimited] ☐ same as report ☐ DTIC users	21 Abstract Security Classification Unclassified	
22a Name of Responsible Individual R. R. Read	22b Telephone *(include Area code)* (408) 646-2382	22c Office Symbol 55Re

Monitoring the Calibration of a Torpedo Test Range

by

Sukru Korlu
Lieutenant J.G. , Turkish Navy
B.S. , Turkish Naval Academy, 1982

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1988

Author: [illegible]
Sukru Korlu

Approved by: R R Read
R. R. Read, Thesis Advisor

[illegible]
H. Larson, Second Reader

[illegible]
P. Purdue, Chairman,
Department of Operations Research

[illegible]
Kneale T. Marshall,
Dean of Information and Policy Sciences

ABSTRACT

Post-test analysis of vehicle tracking data taken from a three dimensional underwater range requires the splicing together of several pieces of track individually produced by distinct transducer arrays. The locations and orientations of the arrays must be known precisely in order to convert locally determined tracks into a coherent record in the general range coordinate system. The maintenance of calibration of the system is a problem and this study presents some least squares methodology that uses the tracking data itself to monitor it. More specificially, methodologies are developed to improve upon the array displacement and orientation correction algorithms and to estimate timing synchronization offsets. Applications are made to real data.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION AND BACKGROUND

A. BACKGROUND

Torpedoes are tested for proper operation at the Dabob Bay and Nanoose underwater ranges operated by the Naval Undersea Warfare Engineering Station, Keyport, Wa. (NUWES). There, the torpedo is tracked by a system of hydrophone arrays on the ocean floor. In order to determine if the torpedo is acquiring and homing on its target properly, accurate tracking is essential.

The arrays in the system are of the short base line type, each containing four sonar transducers placed rigidly at the corners of a cube in a manner that describes a cartesian coordinate system in three dimensions (See Figure 1). An array receives a distinctive signal from a synchronously timed pinger attached to the torpedo. The differentials of the sound wave front times of arrival at the four hydrophones allow the computation

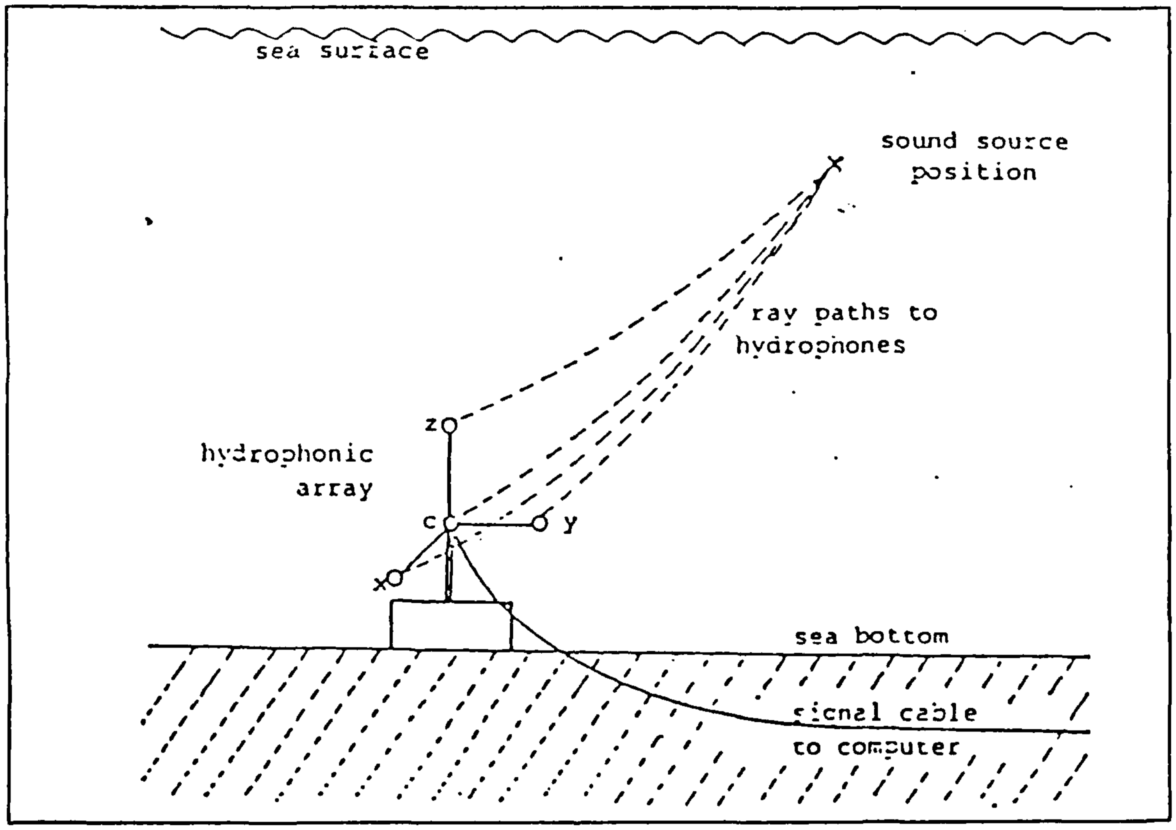

Figure 1. Schematic diagram of three-dimensional hydrophonic array.

of the azimuth and elevation angles of the normal to the wave front at the origin of the local coordinate system. Then assuming direct path propagation, one can ray trace using Snell's law [Ref. 1], starting with the aforementioned elevation angle and utilizing a velocity-versus-depth profile for the speed of sound in the water. Finally, the time differential between the source pulse at the torpedo and its arrival at the array is used to stop the ray-tracing algorithm and determine the location of the torpedo relative to the array. The local track is the sequential set of these estimated positions.

The functioning of the Range's tracking system requires knowledge of the location and orientation of each array. Moreover once these values are established, they must be monitored against slippages of various forms, i.e., calibration must be maintained. Also, the synchronization of the pinger with the shore based computers requires great precision , because timing errors can be confused with other calibration errors.

B. INTRODUCTION

Each array in the range system operates over a limited radius. As the target passes through the range, it is tracked at discrete time points (called point counts) by a number of these arrays (See Figure 2 on page 3). The overall path is constructed by transforming each piece of local track to the coordinates of the range based upon the assumed location and orientation of the various local coordinate systems. Discontinuities, or mismatches occur because the track produced by one array does not mesh exactly with that produced by a neighboring array. If the tracking were free of noise, but the actual positions or orientations of the arrays were different from the ones assumed, errors in the two versions of the track would be obvious and interpretable.

A second possible source of error is due to the water conditions. The ray-tracing procedure depends on the velocity of sound in each layer of water. A measurement of the sound velocity, from the surface to the bottom, is taken for each day's operation. This information, called a depth-velocity profile, is assumed to be constant throughout the range for that day's operations. The velocity of sound in water is highly variable, and influenced by depth, temperature, and salinity changes. Any errors, either a bias of the measurement or inhomogeneity from one part of the range to another, will affect the accuracy of the range. Also any noise in the water will influence the accuracy of the range.

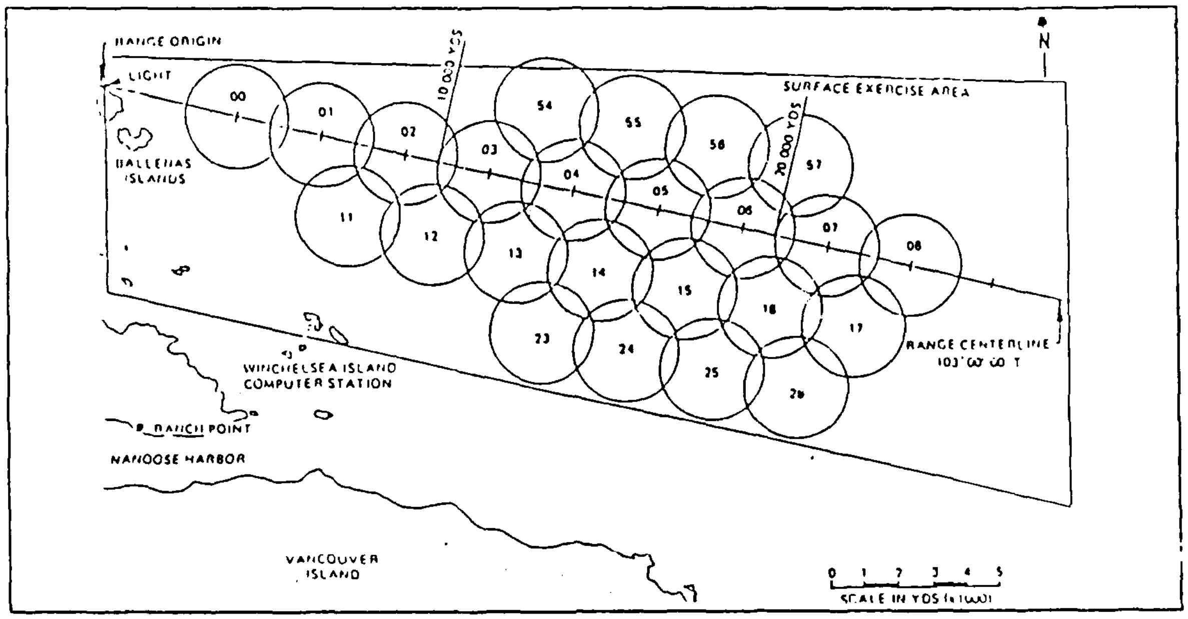

Figure 2. **Plan view of range coordinate system and array tracking regions.**

The purpose of this thesis is to study the use of crossover data (i.e., two versions of a track for the same point count set when the vehicle is in the range of two or more different arrays) for monitoring the calibration of the torpedo test range. A major goal is to address the question of whether mismatches in the different array tracks can be attributed to slippages in array position and orientation, timing errors or whether some other source of systematic error should be treated.

C. STRUCTURE

The structure of the remainder of the thesis is as follows: Chapter 2 investigates the source of error in the array locations. Chapter 3 deals with the timing offset error between the torpedo pinger and the shore computer. Chapter 4 gives a methodology in order to combine multiple array displacement and orientation corrections into a single correction. Conclusions and recomendations are provided in chapter 5.

II. ARRAY DISPLACEMENT AND ORIENTATION CORRECTIONS

A. MODEL

Consider a set of point counts S in a crossover region. It is convenient to refer to the two arrays as the "left" array and the "right" array. Let the crossover data in range coordinates be

$$Y(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{bmatrix} \quad \textit{for track determined by the } \textbf{left} \textit{ array}$$

and

$$X(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} \quad \textit{for track determined by the } \textbf{right} \textit{ array} \qquad \textit{for} \quad t \in S$$

T = *Number of points in S*

provided by two different sensing arrays. Let us agree that the Y(t) data comes from the array whose location and orientation are established, and our goal is to check the calibration of the other array. In particular, does there exist a 3 vector $A \neq 0$ and a 3×3 orthonormal matrix $B \neq I$ such that the adjusted track values,

$$X'(t) = A + BX(t) \tag{2.1}$$

are in better agreement with the Y(t) than are unmodified X(t) ?

The vector A is related to a displacement of the sensing array and the matrix B is related to a correction of its orientation. If we let α be the location of the array in range coordinates, $\xi(t)$ be the local track determined by the array relative to its own position and orientation, and β be the orthonormal orientation adjustment which rotates the local coordinates into directional alignment with the range coordinates, then

$$X(t) = \alpha + \beta\xi(t) \tag{2.2}$$

and

$$\hat{X}(t) = A + B\alpha + B\beta\xi(t) \tag{2.3}$$

The corrected location and orientation adjustments are $A + B\alpha$ and $B\beta$, respectively.

Estimates of A and B are obtainable using the principle of least squares. We will minimize the average square deviation between Y(t) and A + BX(t) for each point count. Using the squared norm notation, define an objective function

$$Q = \underset{t}{\text{ave}} \; \| Y(t) - A - BX(t) \|^2$$

i.e.,

$$Q = \frac{1}{T} \sum_{t \in S} \left\{ \sum_{j=1}^{3} \left[y_j(t) - A_j - \sum_{k=1}^{3} B_{jk} x_k(t) \right]^2 \right\} \tag{2.4}$$

There is a constraint that accompanies the minimization of Q. Recall that the sensing arrays are rigid local cartesian coordinate systems. If they have slipped (i.e. , moved physically such as by the action of a ship's anchor, hooking a cable) then they undergo a displacement and a reorientation. The matrix B should be orthonormal as is the matrix β in the original calibration, thus our problem can be written as.

$$\min \; Q = \underset{t}{\text{ave}} \; \| Y(t) - A - BX(t) \|^2 \tag{2.5}$$

subject to

$$B^T B = I \tag{2.6}$$

where I is a 3 × 3 identity matrix.

Solution methodology and a program (Keymain) for this procedure are presented in [Ref. 2].

B. CROSSOVER DATA

The crossover data described above forms but a small part of the data records collected during regular testing operations. Our purpose requires software to extract and organize the crossover data portions.

The Keyport torpedo tracking data file (which is also called a T file) is a list of records. Each record includes point count, array number and the position of the torpedo in the three dimensional range coordinate system. There are approximately 3000 records in a file. Most of the records in a file are single zone records. In other words there is only one array that receives the torpedo signal at that moment (point count).

In order to seperate crossover data, that is the group of records taken from two or more different arrays at the same time, we used the program Keygate which was developed under the direction of Prof. Robert R. Read, Naval Postgraduate School, 1985. Fortran 77 code for this program is presented in appendix A. The 21 August 1987 Keyport T file was selected for illustration. The program Keygate was used to produce a set of crossover data, provided by the arrays 3, 4 and 13. Figure 3 on page 7 shows such a set and provides a general plan view in a macro scale. It shows array locations and torpedo tracks from array 3, array 4 and array 13 in the crossover zone. Since the coverage of the plot is very large, it is too difficult to discriminate the three determinations of track. This particular set is a bit rare in that three arrays track the target vehicle for these point counts.

Figure 4 on page 8 gives a more detailed (micro) view for these tracks in the crossover zone. The track plotted with the star symbol belongs to array 3. The other two tracks, which are plotted with plus and circle symbols, are provided by array 4 and array 13 respectively. Mismatches between the tracks can be easily seen in this figure.

C. TEMINOLOGY, NOTATION AND RESULTING CORRECTIONS

Assuming the location and orientation of the array 3 are already established, we can check the relative calibration of array 4 and array 13. Recall that the vector A is related to a displacement of the sensing array and the matrix B is related to a correction of its orientation. The estimated displacement vector (D) can be obtained from (2.3), i.e.,

$$D = A + (B - I)\alpha \tag{2.7}$$

To describe the orientation corrections we use the Euler angle representation of B as developed in [Ref. 2,3]. Let us shorten the writing: let $c_i = \cos\phi_i$ and $s_i = \sin\phi_i$ for i = 1,2,3, (where ϕ_i is the rotation angle for axis i); then the individual planar rotations can be writen as follows:

$$\rho_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_1 & -s_1 \\ 0 & s_1 & c_1 \end{bmatrix} \quad \rho_2 = \begin{bmatrix} c_2 & 0 & -s_2 \\ 0 & 1 & 0 \\ s_2 & 0 & c_2 \end{bmatrix} \quad \rho_3 = \begin{bmatrix} c_3 & -s_3 & 0 \\ s_3 & c_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

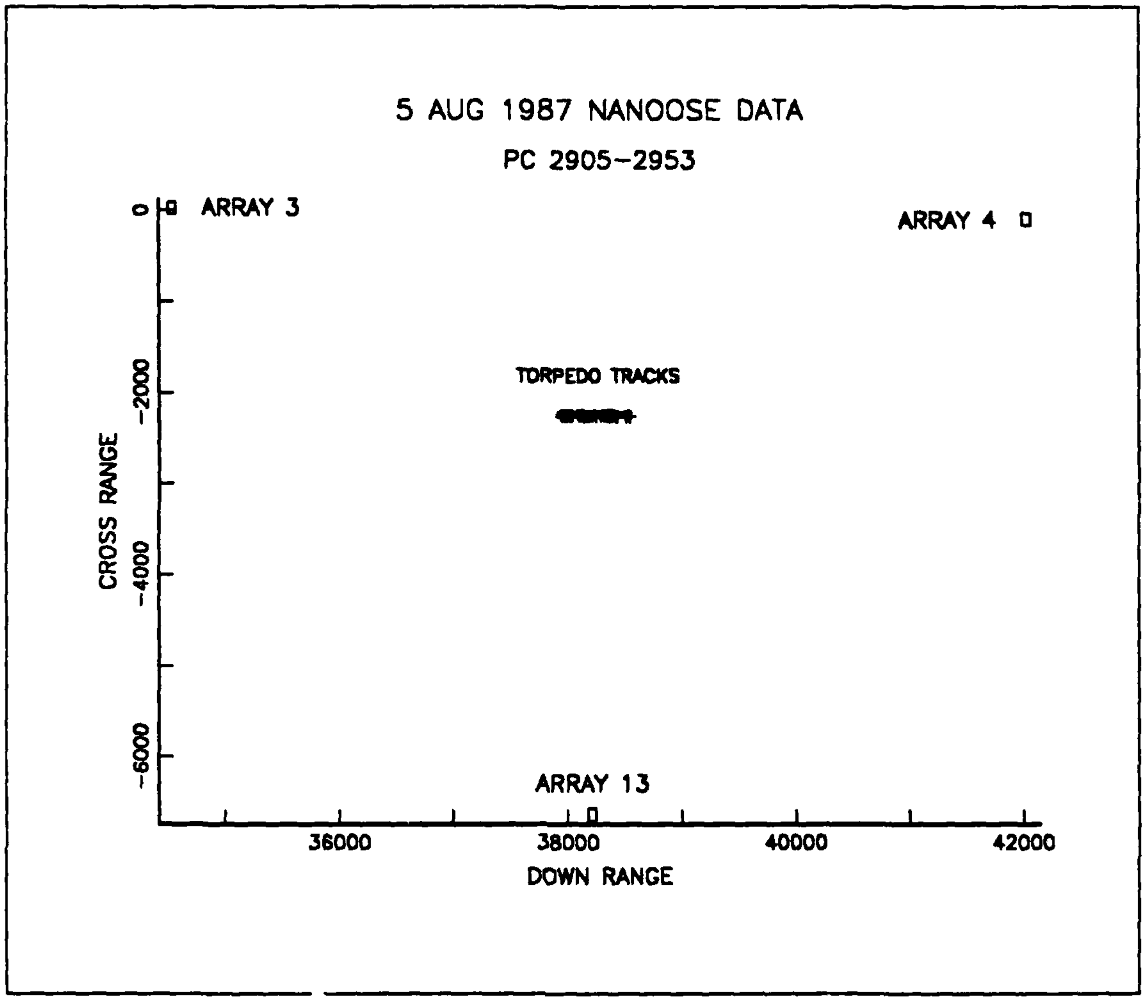

Figure 3. General plan view of the crossover data.

For example, in the plane of $(x_2\ ,\ x_3)$ with x_1 held fixed the effect of ρ_1 is the usual rotation of coordinates [See Figure 5 on page 9].

These three rotation angles when used together describe the three degrees of freedom contained in the matrix B. Specificially,

$$B = \rho_3\rho_2\rho_1 \tag{2.9}$$

In order to present the analysis with simplified summaries, the three components of the displacement (2.7) can be replaced by their magnitude (MD).

$$MD = [\ \| D \|^2]^{1/2} \tag{2.10}$$

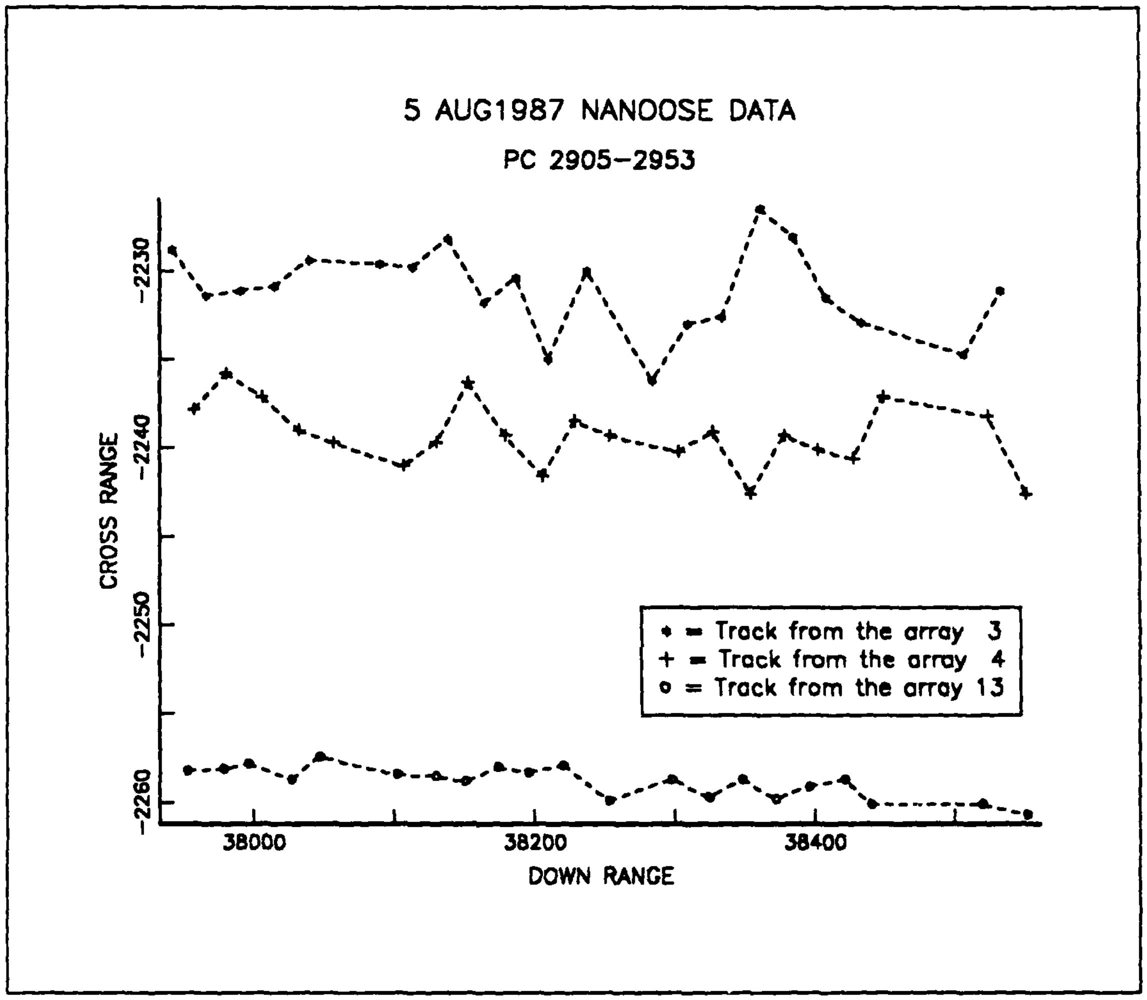

Figure 4. Plan View of the tracks in the crossover zone.

Also the three Euler angles can be replaced by one maximum angle of rotation [See Ref. 3]. We can call the maximum angle of rotation the total rotation.

Using the aforementioned crossover data file with the program Keymain, resulting summary corrections for array 3 and array 13 relative to array 2, are presented in Table 1 on page 9; a "worst case" has been chosen deliberately. In this table QI and QL are the initial and the last values of the least squares objective function, (2.4). Also MD and TR represent the magnitude of displacement and the total rotation respectively.

As can be seen from Table 1 on page 9, corrections for the array locations and orientations are impossibly large. It is well known that least square techniques often over

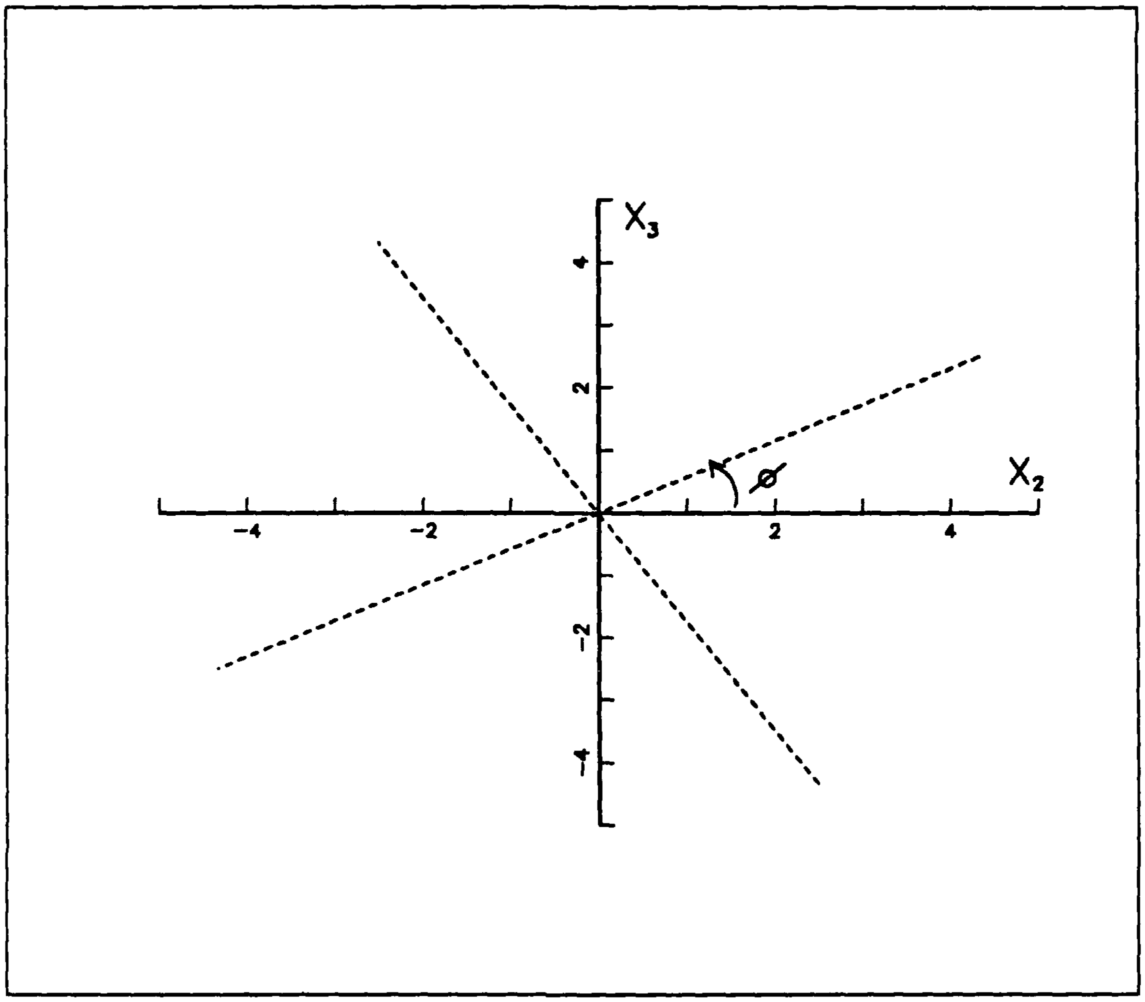

Figure 5. Effect of rotation.

optimize, and such is suspected here. Also we can investigate other sources of error. We will deal with these sources in the following chapters.

Table 1. INITIAL DISPLACEMENT AND ORIENTATION CORRECTIONS

Array	QI (ft^2)	QL (ft^2)	% Reduction	MD (feet)	TR (degree)
4	18.32	16.22	11.04	1577.94	38.52
13	24.90	18.40	26.11	7575.39	115.19

III. TIMING OFFSET ERROR

A. ANALYSIS

Assume there is a timing *synchronization error* between the torpedo pinger and the shore computer. If this error is negative, that is the torpedo pinger sends the sound signal prior to the programmed time, then the calculated distance from the torpedo will be smaller than the actual. If we denote the timing offset error by Δ and the speed of sound at the depth of the torpedo by V(t) for a fixed $t \in S$, the distance between the actual and the observed position of the target will be $\Delta V(t)$. If the timing offset error is positive then the calculated distance will be $\Delta V(t)$ larger than the actual , but in either case, there will be no error in the ray trace path direction in the vertical plane of the array and the torpedo.

A timing offset error will cause the track determined by a given array to shift (in the horizontal) along the direction of a straight line from the array to the target for each point count. To illustrate, without cluttering the diagram, four point counts are selected and these direction lines are drawn for them using each of three determinations of track by the three arrays in Figure 6 on page 11. Note that if the target determinations are stretched along these directions, the overall coherence of the three versions of track can be improved. There will also be some stretching in the vertical. Although it is nonlinear the first order approximation will be used.

B. METHODOLOGY

1. A model for calculating the timing offset error

Let $X^R(t)$ denote the observed track from one array in the range coordinate system.

$$X^R(t) = \begin{bmatrix} x_1^r(t) \\ x_2^r(t) \\ x_3^r(t) \end{bmatrix} \tag{3.1}$$

If we denote the array location in the range coordinate system by α

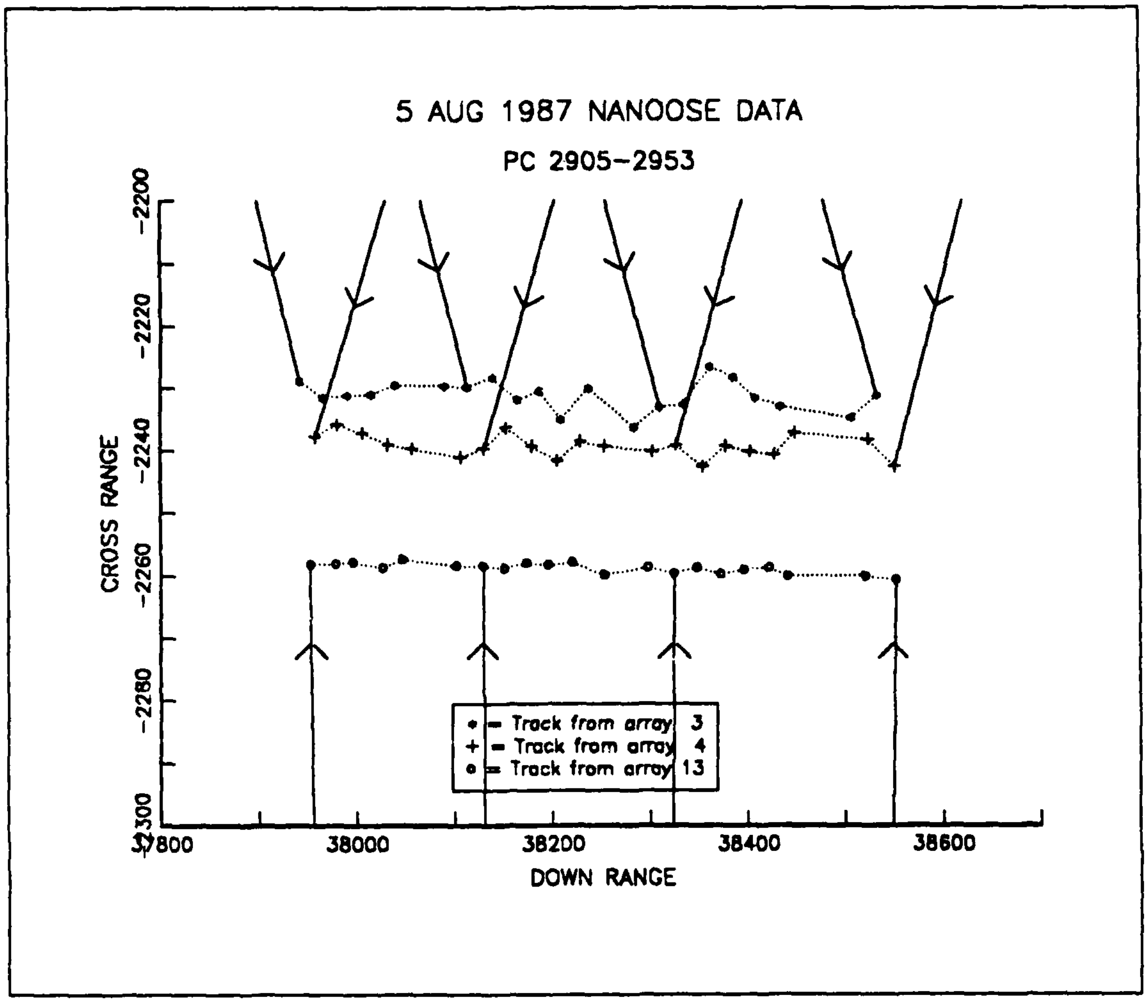

Figure 6. Timing offset error included crossover data

$$\alpha = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \tag{3.2}$$

then the local track (with orientation the same as that of the range), ($X^L(t)$), for this array can be calculated by

$$X^L(t) = \begin{bmatrix} x_1^l(t) \\ x_2^l(t) \\ x_3^l(t) \end{bmatrix} = X^R(t) - \alpha \tag{3.3}$$

Let $\phi(t)$ be the azimuth angle, $h(t)$ be the horizontal range component and $z(t)$ be the vertical component at point count t. Then the postion of the torpedo in this local coordinate system is

$$X^L(t) = \begin{bmatrix} x_1^l(t) \\ x_2^l(t) \\ x_3^l(t) \end{bmatrix} = \begin{bmatrix} h(t)\cos\phi(t) \\ h(t)\sin\phi(t) \\ z(t) \end{bmatrix} \tag{3.4}$$

If there is a timing offset error of size Δ (positive or negative), then the correct local track will be shifted (stretched or contracted) along the ray tracing path. A first order approximation of this shift requires the elevation angle, $\theta(t)$, of the ray trace in the water layer of the observed track. Also we need V(t), the speed of sound in this layer. Using basic trigonometry, we can find the horizontal and vertical components of the shift, $\Delta V(t)\cos\theta(t)$ and $\Delta V(t)\sin\theta(t)$ respectively (See Figure 7 on page 15). Then the corrected horizontal and vertical components are

$$h(t,\Delta) = h(t) + \Delta V(t)\cos\theta(t) \tag{3.5}$$

$$z(t,\Delta) = z(t) + \Delta V(t)\sin\theta(t) \tag{3.6}$$

Using Equation (3.4), we can write the corrected position in the local coordinates.

$$X^L(t,\Delta) = \begin{bmatrix} h(t)\cos\phi(t) \\ h(t)\sin\phi(t) \\ z(t) \end{bmatrix} + \begin{bmatrix} \Delta V(t)\cos\theta(t)\cos\phi(t) \\ \Delta V(t)\cos\theta(t)\sin\phi(t) \\ \Delta V(t)\sin\theta(t) \end{bmatrix}$$

$$X^L(t,\Delta) = X^L + \begin{bmatrix} \Delta V(t)\cos\theta(t)\cos\phi(t) \\ \Delta V(t)\cos\theta(t)\sin\phi(t) \\ \Delta V(t)\sin\theta(t) \end{bmatrix} \tag{3.7}$$

For convenience, let a(t) be the coefficient of the error term in the local coordinates,

$$a(t) = \begin{bmatrix} V(t)\cos\theta(t)\cos\phi(t) \\ V(t)\cos\theta(t)\sin\phi(t) \\ V(t)\sin\theta(t) \end{bmatrix} \tag{3.8}$$

so we can write the track in local coordinates as follows.

$$X^L(t, \Delta) = X^L(t) + \Delta a(t)$$

If we add α to both sides of the above equation, then the track in range corrdinates can be written as follows.

$$X^R(t, \Delta) = X^R(t) + \Delta a(t) \tag{3.9}$$

where $X^R(t)$ is the corrected track in the range coordinates. If we use the notation $Y^R(t, \Delta)$ for the track from another array,

$$Y^R(t, \Delta) = Y^R(t) + \Delta b(t) \tag{3.10}$$

where b(t) is the coefficient of the error term for the second array. Corrected tracks in the range coordinate system from both arrays are given by (3.9) and (3.10).

Now we can calculate the timing offset error by minimizing the following least squares objective function.

$$\min_{\Delta} \sum_t \| X^R(t, \Delta) - Y^R(t, \Delta) \|^2 \tag{3.11}$$

or

$$\min_{\Delta} \sum_t \| [X^R(t) + \Delta a(t)] - [Y^R(t) + \Delta b(t)] \|^2 \tag{3.12}$$

After taking the derivative of the least squares objective function with respect to Δ and equating to zero, we can find an explicit solution for Δ.

$$\Delta = -\frac{\sum_t [X^R(t, \Delta) - Y^R(t, \Delta)]^T [a(t) - b(t)]}{\sum_t [a(t) - b(t)]^T [a(t) - b(t)]} \tag{3.13}$$

In order to see the result is a minimum, one can easily take the second derivative with respect to Δ and check that it is larger than zero.

For calculating a(t) and b(t) we have to find V(t). It can be found from the depth-velocity profile . Also $\phi(t)$ can be calculated from Eq. (3.4)

$$\phi(t) = \tan^{-1} \frac{x_2^{r\prime}(t)}{x_2^{r\prime}(t)} \tag{3.14}$$

The problem here is to find the elevation angle at the depth of torpedo ($\theta(t)$). In order to find it, one has to use a ray tracing algorithm.

2. Raytracing

a. Background

We will use a raytracing algorithm in order to find the elevation angle at the depth of torpedo. There are two general models used in raytracing. Both methods involve dividing the water into relatively thin layers, and then modelling the ray path through each succesive layer.

The first method is ISOSPEED. In this model, the speed of sound is assumed to be constant in each layer. The velocity chosen is the mean velocity for the layer. The ray path is decribed from layer to layer by repeated application of Snell's law.

The second method is ISOGRADIENT. In this model the velocity of sound in water assumed to be linear and described by the follwing equation.

$$V_z = V_0 + V_1 \times Z \tag{3.15}$$

where V_0 is the speed of sound at the water surface, Z is the depth, and V_1 is the gradient. The gradient represents the rate of change in velocity as the depth increases. When using a model with a single layer, V_0 and V_1 are determined by least squares regression of velocity and depth. In the multilayer model, V_0 is the velocity extrapolated to the surface ($Z < 0$), and V_1 is the gradient within that layer.

The testing of both approaches proved this to be an extremely delicate calculation. An algorithm using the isospeed method was less analytically complex and computationally less intensive, but less accurate than the isogradient case. Therefore we will consider only isogradient raytracing.

b. Isogradient Raytracing

To begin a discussion of isogradient raytracing, assume that we are dealing with only one layer, and that the velocity of sound in that layer is given by Eq. (3.15). If we denote the position of the array and the position of the torpedo by (A_1, A_2) and (P_1, P_2) in the vertical plane that contains the array and the torpedo, then the ray path is a circular arc in that plane with center (C_1, C_2). We need equations to find the center from the given parameters. C_2 is given by the following equation.

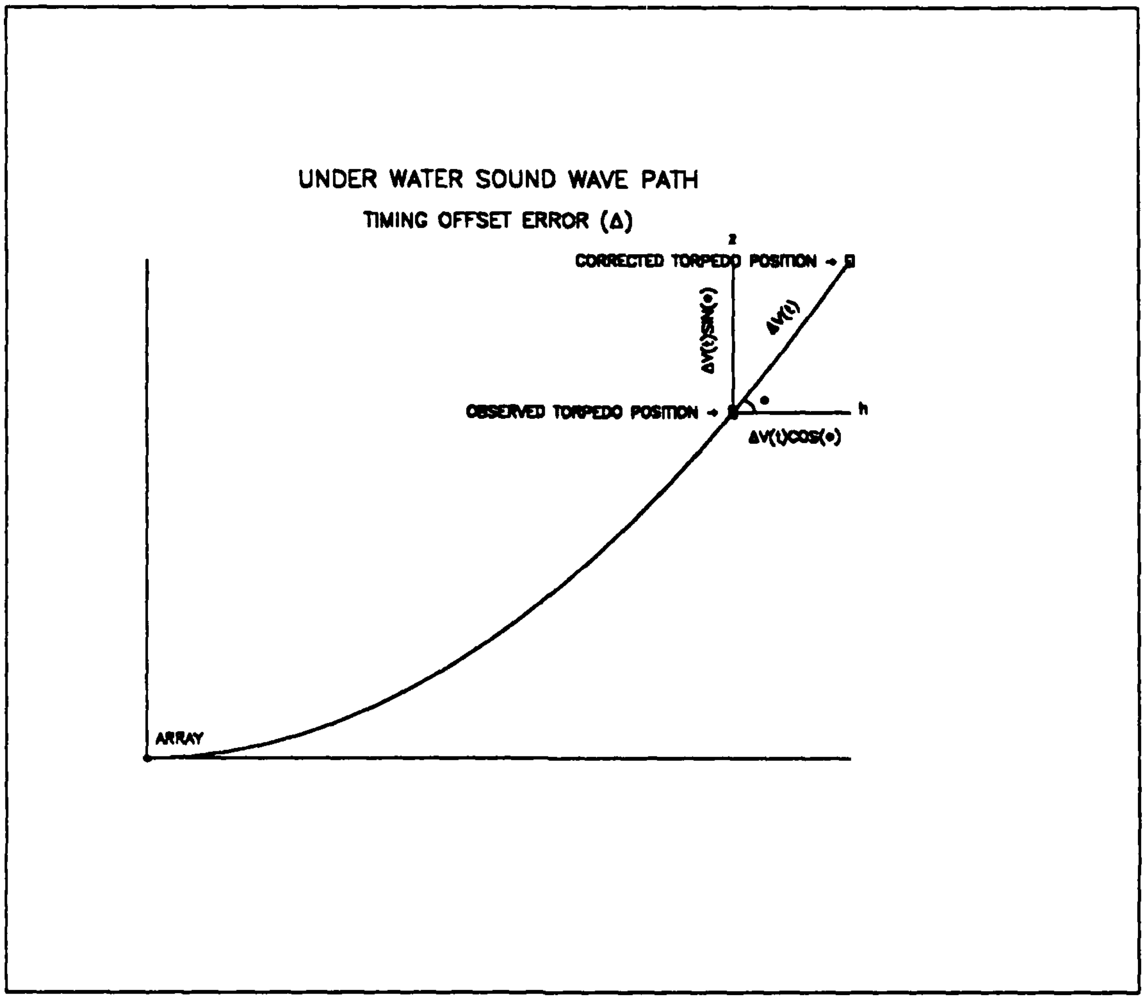
UNDER WATER SOUND WAVE PATH
TIMING OFFSET ERROR (Δ)
z
CORRECTED TORPEDO POSITION →
ΔV(t)SIN(•)
ΔV(t)
•
OBSERVED TORPEDO POSITION →
h
ΔV(t)COS(•)
ARRAY

Figure 7. Sound wave path

$$C_2 = -\frac{V_0}{V_1} \tag{3.16}$$

To find C_1, let R be the length of the radius of the circle. Calculating the distance yields the following equations.

$$(A_1 - C_1)^2 + (A_2 - C_2)^2 = R^2 \tag{3.17}$$

$$(P_1 - C_1)^2 + (P_2 - C_2)^2 = R^2 \tag{3.18}$$

Combining these two equations to solve for C_1 yields the following.

$$C_1 = \frac{P_1 + A_1}{2} + \frac{P_2 - A_2}{2(P_1 - A_1)}(P_2 + A_2 - 2C_2) \tag{3.19}$$

Using trigonometric substitions, C_1 can also be expressed by following equation.

$$C_1 = \frac{(C_2 - A_2)}{\tan\theta} \tag{3.20}$$

where θ is the elevation angle at the array, measured from the horizontal.

The sound velocity does not remain constant along this arc. Therefore, the velocity must be integrated with respect to distance along the ray path to determine the time. If we let θ_0 be the elevation angle of the ray at the torpedo, and θ_1 be the elevation angle of the ray at the hydrophone, then the time of transit T can be expressed by the following equation.

$$T = \frac{1}{V_1}\ln\frac{\cos\theta_1(1+\sin\theta_0)}{\cos\theta_0(1+\sin\theta_1)} \tag{3.21}$$

Other quantities needed can be easily obtained by trigonometric substition into these equations [See Ref. 4].

c. Multilayer Raytracing

In actuality, the isogradient approach is an approximation because the gradient is not linear. Needed accuracy is obtained by dividing the water into 25 foot thick layers. Starting at the ocean surface and going down to the ocean floor. The speed of sound is measured every 25 feet. The difference in sound velocity from one measurement to another is used to determine a velocity gradient for each layer. The isogradient approach is then used within each layer. Starting with a guess for the elevation angle at the array, using single layer approximation,

$$\theta_1 = \tan^{-1}\frac{A_1 - C_1}{A_2 - C_2} \tag{3.22}$$

we can calculate the ray invariant (RV). Note that, the ray invariant is the same for all layers.

$$RV = \frac{\cos\theta_1}{V_j} \tag{3.23}$$

where V_j is the speed of sound at the depth of the array.

Let I be the layer that contains the target and J be the layer that contains the array. If we arrange layer limits, such that the target depth will be the upper limit for layer I and the depth of the array will be the lower limit for layer J, then the elevation angles at layer I and at layer J will be equal to the elevation angles at the target and at the array respectively. Note that, all the layers between I and J have the thickness of 25 ft. But this is not guaranteed for layer I and layer J.

In order to calculate elevation angle at the depth of the target we will use a recursive algorithm. First, using ray invariant we can calculate the elevation angle in each layer k from I to J.

$$\theta_k = \cos^{-1}[RV \times V_k] \tag{3.24}$$

where V_k is the speed of sound in layer K. Then using single layer isogradient approximation, we can calculate the horizontal distance (R_k) in each layer, beginning from layer J through layer I. Summing the horizontal distances in each layer from I to J, gives the horizontal distance of the torpedo from the array.

$$R = \sum_{k=I}^{J} R_j \tag{3.25}$$

If this distance is in the tolerance limits of the observed horizontal range, we can stop the algorithm. Otherwise we have to update our initial guess for the elevation angle at the array by the following formula,

$$\theta_{1_{new}} = \tan^{-1}\left[\tan \theta_{1_{old}} \frac{R}{P_1} \right] \tag{3.26}$$

and calculate the horizontal range until reaching the tolorance limits of observed range. When we find the horizontal range from the algorithm within the tolerance limits of the observed horizontal range, the last calculated elevation angle in layer J is the elevation angle we desired [See Ref. 5].

3. Program TOE

The author's Fortran 77 program to implement the procedure for finding and eliminating the timing offset error, is presented in appendix B. It uses the crossover data from the output of program Keygate. Other inputs are the depth-velocity profile and the range configuration file.

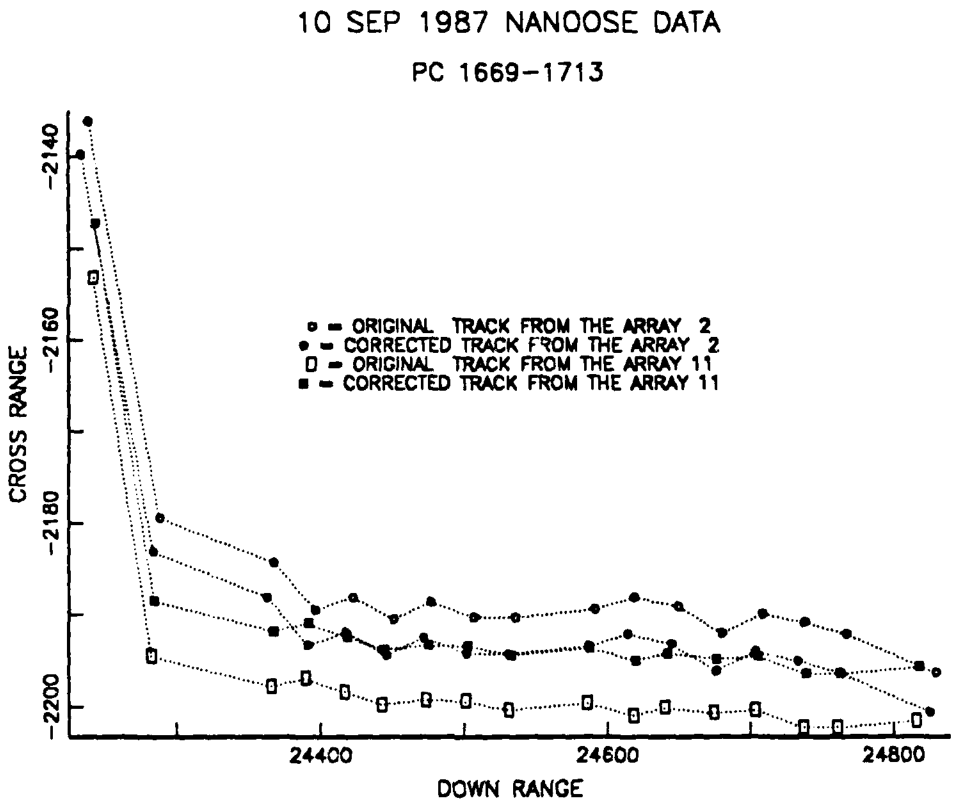

Figure 8. Plan view for the original and corrected tracks

The main program calls five subroutines. Subroutine Local finds the local track, than subroutine Transfrm transforms the local track to its azimuth, horizontal and vertical components. Subroutine Lsline calculates the speed of sound at the water surface and the gradient, using single layer approximation. Subroutine Rtrace finds the elevation angle at the torpedo using the multilayer isogradient raytracing algorithm. Finally subroutine Cdelta calculates the timing offset error and writes the pair of clean track to a user identified output file. For input crossover data that includes 30 point counts, it takes approximately 20 seconds to run this program on an IBM XT.

4. Result

In order to see the effect of eliminating timing offset error we produced ten other crossover data files using program Keygate. These data files come from three days. We thought that, if there is a timing offset error in the data then it must be the same for each operation (day) and each array. We calculated the timing offset error for each of the ten data files. Data dates and summary results are presented in Table 2 on page 20

It can be seen from Table 2 on page 20 that the timing offset error is fixed for each day. Figure 8 on page 18 gives a plan view for the corrected and uncorrected crossover data. We denote uncorrected data with the file N10S87.001 and corrected data with the file C10S87.001.

We calculated the array displacement and orientation corrections from ten corrected crossover data files. Using array 2 as the base array, we got corrections for the arrays 3 and 11. A summary of these corrections is presented in Table 3 on page 23. As before QI and QL are the initial and the last values of the least squares objective function. Also MD and TR represent the total displacement and total rotation respectively.

It can be seen in Table 3 on page 23, there are still big array displacement and orientation corrections, even though there is no timing offset error in the data. We got large amounts especially in the displacement corrections for the vertical axsis. Since we know that there is no effect in the water that moves the array in the vertical direction, we decided to add another condition to our minimization problem, in order to hold the array at the bottom of the ocean when we get the displacement and orientation corrections.

C. A MODIFICATION TO THE RANGE CALIBRATION METHODOLOGY

1. Model

Recall from chapter 2, our goal is to estimate A and B using the principle of least squares. Our objective function was stated in Eq. (2.4) and Eq. (2.5). Since we decided to hold the array at the bottom, we have to add another constraint to the minimization problem, i.e., $D_3 = 0$.

Our goal is to estimate A and B while minimizing the least square objective function Q, see (2.4),

$$Q = \operatorname*{ave}_{t} \ \| Y(t) - A - BX(t) \|^2 \tag{3.27}$$

subject to the constraints

$BB^T = I$

$D_3 = 0$ (3.28)

where $D^T = (D_1, D_2, D_3)$, see (2.7).

Table 2. TIMING OFFSET ERROR FOR THE CROSSOVER DATA FILES

No	File Name	Date	Left Array	Right Array	Point Count	Timing Error
1	N05A87.001	5 Aug. 1987	2	3	25	-0.003
2	N05A87.002	5 Aug. 1987	2	3	25	-0.003
3	N05A87.003	5 Aug. 1987	2	3	25	-0.003
4	N05A87.004	5 Aug. 1987	2	11	20	-0.003
5	N05A87.005	5 Aug. 1987	2	11	20	-0.003
6	N05A87.006	5 Aug. 1987	2	11	20	-0.003
7	N05A87.007	5 Aug. 1987	2	11	20	-0.003
8	N21A87.001	21 Aug. 1987	2	3	23	-0.001
9	N21A87.002	21 Aug. 1987	2	3	26	-0.001
10	N10S87.001	10 Sep. 1987	2	11	17	-0.001

Now let us make a change and work with the deviations.

$Y_d(t) = Y(t) - \bar{y}$ (3.29)

$X_d(t) = X(t) - \bar{x}$ (3.30)

where $\bar{x} = \underset{t}{\text{ave}}\, X(t)$ and $\bar{y} = \underset{t}{\text{ave}}\, Y(t)$. Then we can write the objective function (3.27) as follows.

$$Q = \underset{t}{\text{ave}} \ \| Y_d(t) + \bar{y} - A - B(X_d(t) + \bar{x}) \ \|^2 \tag{3.31}$$

or

$$Q = \underset{t}{\text{ave}} \ \| [Y_d(t) - BX_d(t)] + [\bar{y} - A - B\bar{x}] \ \|^2 \tag{3.32}$$

It can be shown (Pythagorean theorem) that, (3.32) is equal to the following.

$$Q = \underset{t}{\text{ave}} \ \| Y_d(t) - BX_d(t) \ \|^2 + \ \|\bar{y} - A - B\bar{x} \ \|^2 \tag{3.33}$$

Also from (2.7) we get $A = D - (B - I)\alpha$. If we use notation Q_o for orientation and Q_l for location,

$$Q_o = \underset{t}{\text{ave}} \ \| Y_d(t) - BX_d(t) \ \|^2 \tag{3.34}$$

$$Q_l = \ \|\bar{y} - A - B\bar{x} \ \|^2 = \ \|\bar{y} - \alpha - D - B(\bar{x} - \alpha) \ \|^2 \tag{3.35}$$

then we can write the objective function as the following.

$$Q = Q_o + Q_l \tag{3.36}$$

In order to minimize Q we should minimize Q_o and Q_l. Now let the covariance matrices be

$$C_{xx} = \underset{t}{\text{ave}}[X_d^T(t)X_d(t)]$$

$$C_{yy} = \underset{t}{\text{ave}}[Y_d^T(t) Y_d(t)]$$

$$C_{yx} = \underset{t}{\text{ave}}[Y_d^T(t)X_d(t)] \tag{3.37}$$

since $B^T B = I$, Q_o can be written as the following [See Ref. 2].

$$Q_o = tr\{C_{yy} - 2C_{yx}B + C_{xx}\} \tag{3.38}$$

where tr is the trace operator. Since the trace is a linear operator, minimizing Q_o is the same as maximizing the following:

$$\max_B (C_{yx}B^T) \tag{3.39}$$

A solution methodology for this problem is presented in [Ref. 2]. After some algebra Q_l can be written as the following.

$$Q_l = \|\bar{y} - \alpha - B(\bar{x} - \alpha)\|^2 - 2D^T[\bar{y} - \alpha - B(\bar{x} - \alpha)] + D^T D \tag{3.40}$$

Taking the derivative with respect to D [See Ref. 3; (eq. 1)] and equating to zero provides an estimator for D (3.41) which minimizes Q_l.

$$\hat{D} = [\bar{y} - \alpha - B(\bar{x} - \alpha)] \tag{3.41}$$

Since D_3 must equal zero (the array does not leave the bottom), we can modify (3.41) to give the following.

$$D = \begin{bmatrix} \bar{y}_1 - \alpha_1 - \sum_{j=1}^{3} B_{1j}(\bar{x}_j - \alpha_j) \\ \bar{y}_2 - \alpha_2 - \sum_{j=1}^{3} B_{2j}(\bar{x}_j - \alpha_j) \\ 0 \end{bmatrix} \tag{3.42}$$

2. Solution Algorithm

In order to solve the problem initialize A and B as follows.

$$B = I$$

$$D = \begin{bmatrix} \bar{y}_1 - \alpha_1 - \sum_{j=1}^{3} B_{1j}(\bar{x}_j - \alpha_j) \\ \bar{y}_2 - \alpha_2 - \sum_{j=1}^{3} B_{2j}(\bar{x}_j - \alpha_j) \\ 0 \end{bmatrix} \tag{3.43}$$

then solve for B by maximizing the following equation in the usual way [Ref. 2],

$$max\ tr\ [C_{yx}B^T] \tag{3.44}$$

Next calculate D with (3.42) and continue the algorithm by calculating another B. If the difference between the old and new B is large, then turn to the beginning and continue, i.e., calculate another D and from this D calculate another B. Stop the algorithm when D and B stabilize. Since the calculation of B while maximizing Eq. (3.47) requires a similar recursive algorithm, this whole procedure is analytically very complex and computationally very intensive.

An APL code to implement this procedure is presented in appendix C. Corr is the main function, it calculates D, QI and QL, orientation matrix B is calculated by the functions Twodim and Bmax.

3. Resulting Corrections

After correcting the tracks for timing offset errors, we ran the program Keymain again. Then the array displacement and orientation corrections from ten crossover data files are presented in Table 4 on page 25 As before QI and QL are the initial and the last values of the objective function. Δx, Δy and Δz are the displacement corrections (feet). Also Euler angle corrections are denoted by ϕ_1, ϕ_2 and ϕ_3 (radians).

One can easily see in Table 4 on page 25 that the location and orientation corrections are quite small for each array. This certainly improves the stuation depicted in Table 1 on page 9. Note also there are some changes from one observation to another for the same array. Since all changes are very small, we decided that, these corrections are affected by some kind of noise. In order to calculate stable array corrections, in other words to calibrate the range, we need to remove this noise from the array displacement and orientation corrections. We will try to find these stable corrections in the next chapter.

Table 3. DISPLACEMENT AND ORIENTATION CORRECTIONS FOR CORRECTED DATA FILES

File	Array	QI	QL	% Reduction	MD (feet)	TR (degree)
C05A87.001	3	288	22.6	92.7	32.9	0.5
C05A87.002	3	169	20.5	87.8	105.5	1.6
C05A87.003	3	51.8	15.2	70.6	7.2	0.2
C05A87.004	11	36	16.9	53	5503.7	82.0
C05A87.005	11	37.1	12.8	65.4	1752.7	24.1
C05A87.006	11	53.8	13.8	74.3	8390.7	175.7
C05A87.007	11	108	17.5	83.7	5799.6	83.7
C21A87.001	3	166	57.1	65.6	95.5	1.06
C21A87.002	3	295	55.5	81.1	2314.2	141.7
C10S87.001	11	39.6	26.5	33	200.2	2.7

But before going on let us gather together some summaries of the several effects for our ten cases. These are provided in Table 5 on page 25. In this table Raw QI is the average difference between the two versions of uncorrected track:

$$Raw\,QI = \underset{t}{\text{ave}} \; \| Y(t) - X(t) \|^2 \tag{3.45}$$

These figures have the dimension of square feet and are generally quite large. The root mean square values (e.q. $\sqrt{1946.301} = 44.1$ feet for the first one) provide an initial raw figure for the seperation of the two versions of track. The Raw QIM column is the mean square deviation if displacement corrections are made without any array orientation changes. These values provide substantial drops from the Raw QI values. If, in addition to displacement corrections, we also make orientation corrections then further improvement ensues, but not much, as shown by the Raw QL column.

Next let us see what can be accomplished when one first corrects the tracks for timing offset errors. The resulting values of (3.45) appear in the Cor QI column. (See also QI in Table 3). Clearly these values represent a considerable improvement over the Raw QI values. When displacement and orientation corrections (program Keymain) are applied to the timing corrected data, the resulting mean square deviation values appear in the Cor QL column. These values compare with the Raw QL values and, like those values, represent displacement and orientation corrections that often are impossibly large. Finally if we require $D_3 = 0$, i.e., the array can not leave the bottom, the final mean square deviations are marked Cor QF (see also QL in Table 4). The important point is that these values are but modestly larger than the Cor QL values, and at the same time represent but small array corrections as evidenced in Table 4. Thus the combination of corrections for timing errors followed by array corrections that do not allow the array to leave the bottom provide us with a very tenable set of values for our objective function.

Table 4. FINAL ARRAY DISPLACEMENT AND ORIENTATION CORRECTIONS

File Name	Array	QI	QL	Δx	Δy	Δz	ϕ_1	ϕ_2	ϕ_3
C05A87.001	3	288	24	0.37	0.02	0	-8.8E-3	-6.7E-5	2.7E-4
C05A87.002	3	169	24	0.31	0.01	0	-9.6E-3	-5.7E-5	2.6E-4
C05A87.003	3	51.8	15.5	0.07	8.6E-3	0	5.1E-4	-2.0E-4	2.2E-4
C05A87.004	11	36	34.7	1.5	-0.1	0	-4.4E-4	7.9E-7	-2.2E-5
C05A87.005	11	37.1	14	2.6	-0.2	0	-2.0E-3	-2.3E-6	-2.1E-5
C05A87.006	11	53.8	17.2	2.9	-0.2	0	-2.5E-3	-2.6E-6	-1.6E-5
C05A87.007	11	108	20.5	4.2	-0.4	0	-3.9E-3	-4.8E-7	-9.7E-6
C21A87.001	3	166	78	0.1	-0.02	0	0.01	-3.7E-4	3.1E-4
C21A87.002	3	295	107	-0.4	0.04	0	0.01	-4.7E-4	8.3E-5
C10S87.001	11	39.6	37.6	1.4	0.1	0	-5.8E-4	-5.9E-6	-2.2E-5

Table 5. COMPARISON OF THE SEVERAL EFFECTS

File Name	Raw QI	Raw QIM	Raw QL	Cor QI	Cor QL	Cor QF
05A87.001	1946.306	26.753	23.081	288	22.6	24
05A87.002	2062.929	35.198	20.289	169	20.5	24
05A87.003	2197.208	15.361	15.313	51.8	15.2	15.5
05A87.004	483.882	35.220	16.510	36	16.9	34.7
05A87.005	548.032	14.098	12.607	37.1	12.8	14
05A87.006	630.1885	17.429	14.015	53.8	13.8	17.2
05A87.007	779.075	20.595	17.415	108	17.5	20.5
21A87.001	472.376	74.759	57.225	166	75.1	78
21A87.002	554.389	103.397	55.423	295	55.5	107
10S87.001	183.488	36.821	26.428	39.6	26.5	37.6

IV. COMBINING MULTIPLE ARRAY CORRECTIONS INTO ONE STABLE CORRECTION

A. RANGE CALIBRATION

Final displacement and orientation corrections for array 3 and array 11 are presented in Table 4 on page 25. It can be seen from this table that array corrections are quite small and reasonable, but if we carefully take a look at the table, we can see some small differences between the corrections for the same array. For our purposes array displacement and orientation corrections must be fixed for each array. Since there is no effect in the water to change the array location and orientations in a couple days, there must be some kind of noise which affect the array corrections. If this is true, in other words, if observed corrections include additional noise, we should remove this noise from the corrections in order to find stable array displacement and orientation corrections. Removing this noise leads to our best estimate of stable corrections for each array. We can calibrate the range after finding stable corrections for each array in the range. Our approach is to remove the noise from the corrections by means of a linear mixed model. We will deal with this subject in the following.

B. LINEAR MODEL

1. Model

Let us assume that the observed corrections for an array include two effects. One of them is the stable correction for that array and the other one is a day effect for each day's experimental setup. If we let α_i be the stable array corrections for the array i, β_j be the effect for day j, and y_{ijk} be the kth set of observed corrections for array i from day j, then we can write

$$y_{ijk} = \alpha_i + \beta_j + e_{ijk} \tag{4.1}$$

where e_{ijk} are the random measurement errors,

$$i = 1,2,\ldots,p$$

$$j = 1,2,\ldots,q$$

$$k = 1,2,\ldots,r_{ij} \tag{4.2}$$

where r_{ij} is the number of observations for array i and for day j. Since we assume that, the array effect (β_j) is random, then our model is a mixed effect model.

Since our model is a mixed effects model and our data is unbalanced, i.e., we do not have equal number of observations for each day and for each array, an acceptable method for estimating the variance components of this model is the fitting constants method. It is also known as Henderson's method III [See Ref. 6]. But first we must recognize that our corrections are multivariate.

2. Multivariate Analysis

Since each observation is a six element vector, consisting of six corrections for the array, our analysis will be a multivariate analysis. There will be two covariance matrices to be estimated, namely, the between-group and the within-group covariance matrices. These two covariance matrices are most often estimated by forming a multivariate analysis of variance and equating mean square matrices to their expectations.

Now let us turn to our problem and write our model again.

$$y_{ijk} = \alpha_i + \beta_j + e_{ijk}$$

$$i = 1,2,\ldots,p$$

$$j = 1,2,\ldots,q$$

$$k = 1,2,\ldots,r_{ij} \tag{4.3}$$

where y_{ijk} is a 6×1 vector of observations, α_i is a 6×1 vector of stable corrections for array i, β_j is a 6×1 vector of noise for day j and e_{ijk} is a 6×1 vector of standard errors. We assume that the β_j are independently and identically distributed $(0, \Sigma_\beta)$ random vectors which are independent of e_{ijk}. The $\{e_{ijk}\}$ are assumed to be independently and identically distributed $(0, \Sigma_e)$ random vectors. Using a modified fitting-of-constants method [See Ref. 7] we can obtain estimators for the between-group covariance matrix (MS_b) and the with-in-group covariance matrix (MS_w), which are approximately unbiased,

$$\hat{\Sigma}_\beta = c^{-1}(MS_b - MS_w) \tag{4.4}$$

$$\hat{\Sigma}_e = MS_w \tag{4.5}$$

where

$$c = (q-1)^{-1}\left\{\left(\sum_{i=1}^{p}\sum_{j=1}^{q} r_{ij}\right) - \left(\sum_{i=1}^{p}\sum_{j=1}^{q} r_{ij}\right)^{-1}\left(\sum_{i=1}^{p}\sum_{j=1}^{q} r_{ij}^{2}\right)\right\} \tag{4.6}$$

If we let

$$\bar{y}_{...} = \left(\sum_{i=1}^{p}\sum_{j=1}^{q} r_{ij}\right)^{-1}\sum_{i=1}^{p}\sum_{j=1}^{q}\sum_{k=1}^{r_{ij}} y_{ijk} \tag{4.7}$$

and

$$\bar{y}_{.j.} = \left(\sum_{i=1}^{p} r_{ij}\right)^{-1}\sum_{i=1}^{p}\sum_{k=1}^{r_{ij}} y_{ijk} \tag{4.8}$$

then the estimators for the between-group covariance matrix (MS_b) and the with-in-group covariance matrix MS_w are the following

$$MS_b = (q-1)^{-1}\sum_{j=1}^{p}(\bar{y}_{.j.} - \bar{y}_{...})(\bar{y}_{.j.} - \bar{y}_{...})^T \tag{4.9}$$

$$MS_w = \left\{\sum_{i=1}^{p}\sum_{j=1}^{q} r_{ij} - [(q-1)+(p-1)]\right\}^{-1}\sum_{i=1}^{p}\sum_{j=1}^{q}\sum_{k=1}^{r_{ij}}(y_{ijk} - \bar{y}_{.j.})(y_{ijk} - \bar{y}_{.j.})^T \tag{4.10}$$

Both matrices from these estimators are nonnegative definite, but this is not guaranteed for the difference. There is a high probability that the result of Eq. (4.4) is not nonnegative definite and it is not a proper covariance matrix. An estimation methodology which produces nonnegative definite covariance matrices is presented in Ref. [8].

If $MS_b - MS_w$ is not nonnegative definite, then in order to get a nonnegative definite estimator for the covariance matrix Σ_β, let $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_6$ be the ordered roots of the following equation.

$$|MS_b - \lambda MS_w| = 0 \tag{4.11}$$

The roots λ_i are called the characteristic roots of MS_w in the metric of MS_b. In order to find λ_i, let us define a matrix L satisfying following equation.

$$L^T MS_w L = I \tag{4.12}$$

We can find a matrix L satisfying (4.12) by Cholesky decomposition. Another choice for L satisfying (4.12) is

$$L = MS_w^{-0.5} = Q_w \, diag\,\{v_1^{-0.5}\; v_2^{-0.5} \ldots v_6^{-0.5}\}\; Q_w^T \tag{4.13}$$

where v_i are the eigenvalues of MS_w and Q_w is the matrix of the corresponding orthonormal eigenvectors of MS_w. With any choice of L satisfying (4.12), the roots λ_i are the eigenvalues of a symmetric matrix $L^T MS_b L$ Now define a matrix P by

$$P = L^{-1} Q \tag{4.14}$$

where Q is the matrix of the corresponding orthonormal eigenvectors of $L^T MS_b L$.

If $\lambda_i \geq 1$ for all i= 1,2,...,6 then $(MS_b - MS_w)$ is nonnegative definite and the estimator (4.4) is in the parameter space. If $\lambda_i < 1$ for some i, then $(MS_b - MS_w)$ is not nonnegative definite. Let k be the number of λ_i such that $\lambda_i \geq 1$. Furthermore let Λ_k = $diag\,\{\lambda_1, \lambda_2, \ldots, \lambda_k\}$, Λ_l = diag $\{\lambda_{k-1}, \lambda_{k-2}, \ldots, \lambda_6\}$ and $P = (P_k, P_l)$, where P_k is $6 \times k$. Then we can write,

$$MS_b - MS_w \;=\; P_k(\Lambda_k - I_{kk})P_k^T + P_l(\Lambda_l - I_{ll})P_l^T \tag{4.15}$$

$$MS_b - MS_w \;=\; (MS_b - MS_w)_+ + (MS_b - MS_w)_- \tag{4.16}$$

Hence by using the metric of MS_w, we have written $(MS_b - MS_w)$ as the sum of a nonnegative definite matrix $(MS_b - MS_w)_-$ and a negative definite matrix $(MS_b - MS_w)_-$. The $(MS_b - MS_w)_-$ can be considered to be the closest matrix to $(MS_b - MS_w)$ among all nonnegative definite matrices that have the same characteristic vectors in the metric of MS_w as those of $(MS_b - MS_w)$. In this sense we can interperet the term $(MS_b - MS_w)_-$ as the nonnegative definite portion of $(MS_b - MS_w)$ obtained by projecting $(MS_b - MS_w)$ in the metric of MS_w on to the set of all nonnegative definite matrices. Thus it is natural to consider $(MS_b - MS_w)_-$ as an estimator of $c\Sigma_e = E[MS_b - MS_w]$ and the remaining part of MS_b

$$MS_b - (MS_b - MS_w)_+ \tag{4.17}$$

as an estimator of Σ_e. An intuitive estimator of Σ_e is a weighted average of the quantity in (4.17) and MS_w

Based on the foregoing discussion, we define estimators of Σ_β and Σ_e in model (4.1) by

$$\Sigma_\beta = 0 \qquad \text{if } k = 0$$

$$\Sigma_\beta = c^{-1}P_k(\Lambda_k - I_{kk})P_k^T \qquad \text{if } 1 \le k \le 6$$

and

$$\Sigma_e = [(q-1)+z]^{-1}\{(q-1)[MS_b - c\hat{\Sigma}_\beta] + zMS_w\}$$

where

$$z = \left[\sum_{i=1}^{p}\sum_{j=1}^{q} r_{ij}\right] - [(p-1)+(q-1)] \tag{4.18}$$

The author's APL code to implement this procedure are presented in appendix D. This program calls the APL eigenvalue library function Eigenr in order to calculate eigenvalues and eigenvectors of a real matrix.

3. Multivariate Estimation.

The estimation of the fixed effects requires values for Σ_β and Σ_e. We will use the estimators developed in the previous section. The least squares estimator for α requires the variance of the observations. To develop this we change notation. Let N be the total number of experiments (i.e., vector observations), then the model can be written as

$$Y = X\alpha + Z\beta + e$$

where

$$Y = [y_{ij}] \qquad i = 1,\ldots,N \;\; j = 1,\ldots,6$$

$$X = [x_{ik}] \qquad i = 1,\ldots,N \;\; k = 1,\ldots,q$$

$$Z = [z_{il}] \qquad i = 1,\ldots,N \;\; l = 1,\ldots,p$$

$$e = [e_{ij}] \qquad i = 1,\ldots,N \;\; j = 1,\ldots,6$$

$\alpha = [\alpha_{kj}] \qquad k = 1,\dots,p \;\; j = 1,\dots,6$

$\beta = [\beta_{lj}] \qquad l = 1,\dots,q \;\; j = 1,\dots,6$

$$N = \sum_{i=1}^{p} \sum_{j=1}^{q} r_{ij}$$

x_{ik} is one if the ith observation provides a correction vector for the ith array, zero otherwise and z_{il} is one if the ith observation was collected on the lth day, zero otherwise. Note that for our data N = 10, p = 2, q = 3. The $\beta_l = (\beta_{l1},\dots,\beta_{l6})$ are independent random vectors for l = 1,...,q with mean zero and covariance matrix Σ_β. The $e_i = (e_{i1},\dots, e_{i6})$ are independent error random vectors for i = 1,...,N with mean zero and covariance matrix Σ_e. All β_l are independent of all e_i. Clearly

$$E[Y] = X\alpha + ZE[\beta] + E[e] = X\alpha \tag{4.19}$$

To obtain covariances it is convenient to change our mental attitude and think of the Y, α, β, e arrays as vectors for which the six values of the second subscript are repeated after each advancement in the first subscript. With this attitude, the model can be described using matrix direct product notation. In what follows liberal use is made of the direct product calculus developed in [Ref. 9, sec. 8.8].

$$Y = (X \otimes I_6)(I_p \otimes \alpha) + (Z \otimes I_6)(I_q \otimes \beta) + e \tag{4.20}$$

where I_p, I_q and I_6 are identity matrices of order p, q and 6 respectively. Now the covariance matrix V of Y can be developed as

$$V = E[Y - (X \otimes I_6)\alpha][Y - (X \otimes I_6)\alpha]^T$$

$$V = E[(Z \otimes I_6)\beta + e][(Z \otimes I_6)\beta + e]^T$$

$$V = E[(Z \otimes I_6)\beta\beta^T(Z^T \otimes I_6)] + E[ee^T]$$

$$V = (Z \otimes I_6)(I_q \otimes \Sigma_\beta)(Z^T \otimes I_6) + (I_N \otimes \Sigma_e)$$

$$V = ZZ^T \otimes \Sigma_\beta + I_N \otimes \Sigma_e \tag{4.21}$$

where I_N is the identity matrix of order N.

The least squares estimator for α is the Aitken estimator

$$\hat{\alpha} = [(X^T \otimes I_6) V^{-1} (X \otimes I_6)]^{-1} (X^T \otimes I_6) V^{-1} Y \tag{4.22}$$

It is useful to use the structure of the experimental design in order to simplify the computation. The components of $X = [x_{ik}]$ are one or zero according to whether the ith experiment involves the kth sonar array, k = 1,..,p. The value of $Z = [z_{il}]$ is one or zero according to whether the ith experiment occurs on the lth day, l = 1,...,q. Note that Z^TZ is a q by q diagonal matrix $(n_1,...,n_q)$ where n_l is the number of experiments included for the lth day. The matrix ZZ^T is a block diagonal matrix $(J_{n_1},...,J_{n_q})$, where J_{n_l} is a square matrix of order n_l and of its components are one. It follows that V is block diagonal with blocks possessing the general form

$$V_n = I_n \otimes \Sigma_e + J_n \otimes \Sigma_\beta$$

Because of this structure, V^{-1} will have a like structure and we need the inverse of individual blocks. This can be accomplished by assuming the form

$$V_n^{-1} = I_n \otimes a + J_n \otimes d_n \tag{4.23}$$

setting

$$[I_n \otimes \Sigma_e + J_n \otimes \Sigma_\beta][I_n \otimes a + J_n \otimes d_n] = I_n \otimes I_6$$

and solving. Thus

$$(I_n \otimes \Sigma_e)(I_n \otimes a) + (J_n \otimes \Sigma_\beta)(I_n \otimes a) + (I_n \otimes \Sigma_e)(J_n \otimes d_n) + (J_n \otimes \Sigma_\beta)(J_n \otimes d_n)$$

$$= I_n \otimes \sigma_e a + J_n \otimes [\Sigma_\beta a + \Sigma_e d_n + n\Sigma_\beta d_n] = I_n \otimes I_6$$

The solution is

$$a = \Sigma_e^{-1} \tag{4.24}$$

$$d_n = -[\Sigma_e + n\Sigma_\beta]^{-1} \Sigma_\beta \Sigma_e^{-1} \tag{4.25}$$

Now, the Aitken estimator for α can be managed through partitioned matrices. The rows of X are partitioned according to $n_1,...,n_q$. Call a typical one X_n. Then, identifying AA from (4.22),

$$AA = (X^T \otimes I_6)V^{-1} = [\ \ldots,(X_n^T \otimes a) + (X_n^T J_n) \otimes d_n, \ldots\ ] \tag{4.26}$$

and the structure of $X_n^T J_n$ is that of a p by n matrix whose kth row is constant and equal to the number of times (experiments) the kth sonar array was included on that particular day.

We also need from (4.22), $BB = (X^T \otimes I_6)V^{-1}(X \otimes I_6)$ whose value can be obtained from (4.26) by summing the quantities of the form $[(X_n^T \otimes a) + (X_n^T J_n) \otimes d_n](X_n \otimes I_6)$. The result is

$$BB = q \sum_{l=1}^{q} X_{n_l}^T X_{n_l} \otimes a \ + \ \sum_{l=1}^{q} X_{n_l}^T J_{n_l} X_{n_l} \otimes d_{n_l} \tag{4.27}$$

which is a square matrix of order 6p. It can be inverted numerically provided p is not very large. Then the Aitken estimator for α can be written as follows.

$$\hat{\alpha} = (BB)^{-1}(AA)Y \tag{4.28}$$

An APL code to implement this procedure for our data is presented in appendix E. This program calls the APL library function Penrose in order to find a generalized inverse of a matrix. Using this program and previous data, we calculated the stable displacement and orientation corrections for the array 3 and for the array 11 relative to the array 2. Resulting displacement and orientation corrections are presented in Table 6.

Table 6. STABLE CORRECTIONS FOR THE ARRAYS RELATIVE TO THE BASE ARRAY.

Array	Δ X	Δ Y	Δ Z	ϕ_1	ϕ_2	ϕ_3
3	0.54881	-0.0500	0	-4.3E-4	-1.9E-5	3.30E-5
11	0.11021	-0.0121	0	0.01400	-9.6E-5	5.03E-5

V. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

As we pointed out at the beginning, the purpose of this thesis is to decide whether or not crossover data can be used to monitor the calibration of the torpedo test range. We used data from the three arrays, using one array as the base array and obtained corrections for the other two arrays. Since the range consists of 24 arrays, the calibrated part of the range is but a small portion.

Although the calibrated part of the range is a tiny portion, corrections for the two arrays gave us an idea for the whole range. Since observed corrections for the third and eleventh array from the uncorrected crossover data, were enormously big, we began to investigate other sources of error. First we calculated the timing offset error by means of Least Squares. Although eliminating the timing offset error did not reduce the amount of corrections as much as expected, we decided that it is creditable, because no matter what array it is, timing offset error was the same for each operation on the same day. Also based upon plots, it is very encouraging to see that the elimination of timing errors reduces the mismatches between tracks from different arrays. This gives an opportunity to correct not only the crossover data but the entire torpedo track.

Next we tried to hold the array at the bottom of the ocean. This reduced the amount of the corrections, also it seems to stabilize the result. After forcing $A_3 = 0$, corrections were quite small but not fixed for each array. In order to get the fixed corrections, we decided to remove the noise which affected the corrections by means of a linear mixed model. We used a fitting of constants method in order to estimate the fixed corrections. Since the resulting corrections were quite small, we decided that the known locations of the two arrays were correct and the source of the mismaches is the timing offset error between the torpedo and the shore computer.

B. RECOMMENDATIONS

In this study we got corrections for two arrays. In order to calibrate the whole range, this methodology must be used with large data files. This requires large computer storage and long CPU time. If the results for the whole range are as good as the results for this small portion, this methodology can be gathered in one computer program. In order to do this some decision rules are required such as what amount of error will be assigned to timing offset error and what amount of error will be assigned to calibration error.

If the ocean bottom is flat, holding the array at the bottom is reasonable. If not, extra work must be done. Because of this, before using our methodology mapping the ocean bottom is required. Also this methodology uses the same base array for each operation. A modification which makes different base arrays acceptable will be useful and maximum likelihood estimation can be used to estimate fixed array corrections. A multivariate version of maximum likelihood estimation is presented in [Ref. 10] and [Ref. 11].

APPENDIX A. PROGRAM KEYGATE

```
C        PROGRAM KEYGATE
C        ***************************************************************
C        ***  Program to read in raw data from Keyport hydrophone     ***
C        ***  arrays, segregate it by mode, and throw out unusable    ***
C        ***  records.  The output of this program is to be read in  ***
C        ***  by the program KEYMAIN.                                 ***
C        ***************************************************************
C
         INTEGER*4 CRT, KBD
         CHARACTER*13 DSNAME, SITNAM
         CHARACTER*9 TEMP1, TEMP2, TEMP3, TEMP4
C
C
         PARAMETER (KBD=5, CRT=6)
C
C
         WRITE(CRT,*) ' Please enter the name of your input file: '
         READ(KBD,'(A)') DSNAME
C
         WRITE(CRT,*) ' Please enter the name of the range',
     +                ' configuration file: '
         READ(KBD,'(A)') SITNAM
C
C
         TEMP1 = 'TEMP1.TMP'
         TEMP2 = 'TEMP2.TMP'
         TEMP3 = 'TEMP3.TMP'
         TEMP4 = 'TEMP4.TMP'
C
C
         CALL STRMOD(CRT,KBD,TEMP1,DSNAME)
C
C
         CALL PAIR(CRT,KBD,TEMP1,TEMP2)
C
C
         CALL RANGE(CRT,KBD,SITNAM,TEMP2,TEMP3)
C
C
         CALL PAIR2(CRT,KBD,TEMP3,TEMP4)
C
C
         CALL REC(CRT,KBD,TEMP4)
C
C
         WRITE(CRT,*)' Operation complete.  KEYGATE terminating... '
C
         STOP
         END
C
```

```
C
C
      SUBROUTINE STRMOD(CRT,KBD,TEMP1,DSNAME)
C     ****************************************************************
C     ** PROGRAM TO STRIP ALL MODES EXCEPT 2'S AND 7'S FROM KEYPORT **
C     ** DATA. (2 INDICATES TARGET SHIP, 7 INDICATES TORPEDO)      **
C     ****************************************************************
C
      CHARACTER DO*2, DSNAME*13, TEMP1*9
      INTEGER PC, ARRAY, CRT, NHEAD
C
      OPEN(1,FILE=DSNAME,STATUS='OLD')
C
C
      WRITE(CRT,*)' How many records of header do you want to',
     +            ' strip off the file?'
      READ(KBD,*) NHEAD
      DO 11 I = 1,NHEAD
      READ(1,*)
   11 CONTINUE
C
      WRITE(CRT,*)' Input mode to be kept?'
      READ(KBD,*) NUM
C
      OPEN(2,FILE=TEMP1,STATUS='NEW')
C
C
 10   READ(1,100,END=50,ERR=40)PC,DO,X,Y,Z,ARRAY,MODE
      IF(DO .NE. '  ')GOTO 20
      IF(MODE .NE. NUM) GOTO 20
      WRITE(2,110)PC,X,Y,Z,ARRAY,MODE
 20   CONTINUE
      GOTO 10
 40   WRITE(CRT,*)' THERE WAS AN ERROR IN THE FILE'
 50   CONTINUE
 100  FORMAT(I5,A2,1X,F7.1,2X,F7.1,2X,F7.1,30X,I2,2X,I1)
 110  FORMAT(1X,I5,2X,3F10.1,2X,I2,2X,I2)
      CLOSE (UNIT=1)
      CLOSE(UNIT=2)
      RETURN
      END
C
C
C
C

      SUBROUTINE PAIR(CRT,KBD,TEMP1,TEMP2)
C     *************************************************************
C     ***  PROGRAM TO PAIR POINT COUNTS AFTER THE DATA HAS BEEN  ***
C     ***  GATED BY " STRMOD ".  SECOND PASS.                    ***
C     *************************************************************
C
C
      DIMENSION X(200), Y(200), Z(200)
      INTEGER*4 PC(200), ARRAY(200), MODE(200), CRT, HOLD
      CHARACTER*9 TEMP1, TEMP2
C
C
C
      OPEN(1,FILE=TEMP1,STATUS='OLD')
      OPEN(2,FILE=TEMP2,STATUS='NEW')
C
      HOLD = 0
      IREC = 0
      NREC = 0
      I = 1
C
C     ...  READ RECORDS BY TWO'S TO COMPARE POINT COUNTS:
C
      READ(1,*,END=40,ERR=30) PC(I), X(I), Y(I), Z(I),
     +              ARRAY(I), MODE(I)
 10   READ(1,*,END=40,ERR=30) PC(I+1), X(I+1), Y(I+1), Z(I+1),
     +              ARRAY(I+1), MODE(I+1)
      NREC = NREC + 1
C
      IF(PC(I+1) .EQ. HOLD) THEN
         WRITE(2,100) PC(I+1), X(I+1), Y(I+1), Z(I+1),
     +                   ARRAY(I+1), MODE(I+1)
         HOLD = PC(I+1)
         GO TO 20
      END IF
C
      IF(PC(I) .EQ. PC(I+1)) THEN
         WRITE(2,100) PC(I), X(I), Y(I), Z(I), ARRAY(I), MODE(I)
         WRITE(2,100) PC(I+1), X(I+1), Y(I+1), Z(I+1),
     +                   ARRAY(I+1), MODE(I+1)
         HOLD = PC(I+1)
         IREC = IREC + 1
         GO TO 20
      END IF
C
      IF(PC(I) .NE. PC(I+1)) THEN
         PC(1) = PC(I+1)
         X(1) = X(I+1)
         Y(1) = Y(I+1)
         Z(1) = Z(I+1)
         ARRAY(1) = ARRAY(I+1)
         MODE(1) = MODE(I+1)
         I = 1
      END IF
C
      GOTO 10
 20   I = I + 1
      GO TO 10
C
C
 30   WRITE(CRT,*)' THERE IS AN ERROR IN THE DATA FILE IN REC.',NREC
      WRITE(CRT,*)' ... OPERATION TERMINATING DUE TO ERROR. '
      STOP
 40     CONTINUE
C
C
        CLOSE(UNIT=1)
        CLOSE(UNIT=2)
C
 100    FORMAT(1X,I5,2X,F7.1,2X,2F9.1,2X,I2,2X,I2)
C
        RETURN
        END
C
C
C
C
C
        SUBROUTINE RANGE(CRT,KBD,SITNAM,TEMP2,TEMP3)
C       ************************************************************
C       ***  This program completes the third gating of Keyport    ***
C       ***  range data.  It reads array location data from a      ***
C       ***  site specific configuration file and tests to see if the***
C       ***  data is in the valid overlap area.                    ***
C       ************************************************************
C
C
        INTEGER*4 ARRAY, CARRAY, CRT, PC
        REAL*4  CONFIG(200,4), LX, LY, LZ, MAXVAL
        CHARACTER SITNAM*13, TEMP2*9, TEMP3*9
C
C
C
C       ...  Open input and output files:
        OPEN(1,FILE=TEMP2,STATUS='OLD')
        OPEN(2,FILE=SITNAM,STATUS='OLD')
        OPEN(3,FILE=TEMP3,STATUS='NEW')
C
C
C       ...  Read site configuration into CONFIG array:
        NREC = 0
        I = 1
 10     READ(2,*,END=40,ERR=30) CONFIG(I,1), CONFIG(I,2), CONFIG(I,3),
     +                          CONFIG(I,4)
        NREC = NREC + 1
        I = I + 1
        GOTO 10
 30     WRITE(CRT,*)' There was an error reading the config',
     +              ' file in record ',NREC
 40     CONTINUE
        CLOSE(UNIT=2)
C
C
        NDREC = 0
C
C       ...  Read X, Y, Z, and ARRAY from input data file:
 45     READ(1,*,ERR=70,END=80) PC, X, Y, Z, ARRAY, MODE
        DO 50 I = 1,NREC
        LX = CONFIG(I,1)
```

```
          LY = CONFIG(I,2)
          LZ = CONFIG(I,3)
          CARRAY = INT(CONFIG(I,4))
          MAXVAL = 4700.
C
C         ...  Match array number in data file with that in config. file.
C         ...  If they are equal compute slant range distance (SRDIST):
          IF(ARRAY .EQ. CARRAY) THEN
             SRDIST = SQRT( (X - LX)**2 + (Y - LY)**2 + (Z - LZ)**2 )
             IF(MAXVAL .GE. SRDIST) THEN
               WRITE(3,100) PC, X, Y, Z, ARRAY, MODE, SRDIST
               NDREC = NDREC + 1
             END IF
          END IF
C
 50       CONTINUE
          GOTO 45
C
 70       WRITE(CRT,*)' There is an error in the data file. '
 80       CONTINUE
C
          CLOSE(UNIT=1)
          CLOSE(UNIT=3)
C
 100      FORMAT(3X,I5,3(3X,F10.1),2(3X,I2),3X,F8.2)
C
C
          RETURN
          END
C
C
C
C
          SUBROUTINE PAIR2(CRT,KBD,TEMP3,TEMP4)
C         ****************************************************************
C         ***  PROGRAM TO PAIR POINT COUNTS AFTER THE DATA HAS BEEN   ***
C         ***  TESTED BY " RANGE ".  FOURTH PASS.                     ***
C         ****************************************************************
C
C
          DIMENSION X(200), Y(200), Z(200), SRDIST(200)
          INTEGER*4 PC(200), ARRAY(200), MODE(200), CRT, HOLD
          CHARACTER*9 TEMP3, TEMP4
C
C
C
          OPEN(1,FILE=TEMP3,STATUS='OLD')
          OPEN(2,FILE=TEMP4,STATUS='NEW')
C
          HOLD = 0
          IREC = 0
          NREC = 0
          I = 1
C         ...  READ RECORDS BY TWO'S TO COMPARE POINT COUNTS:
          READ(1,*,END=40,ERR=30) PC(I), X(I), Y(I), Z(I),
     +                   ARRAY(I), MODE(I), SRDIST(I)
 10     READ(1,*,END=40,ERR=30) PC(I+1), X(I+1), Y(I+1), Z(I+1),
     +               ARRAY(I+1), MODE(I+1), SRDIST(I+1)
        NREC = NREC + 1
C
        IF(PC(I+1) .EQ. HOLD) THEN
           WRITE(2,100) PC(I+1), X(I+1), Y(I+1), Z(I+1),
     +                  ARRAY(I+1), MODE(I+1), SRDIST(I+1)
           HOLD = PC(I+1)
           GO TO 20
        END IF
C
        IF(PC(I) .EQ. PC(I+1)) THEN
           WRITE(2,100) PC(I), X(I), Y(I), Z(I), ARRAY(I), MODE(I),
     +                  SRDIST(I)
           WRITE(2,100) PC(I+1), X(I+1), Y(I+1), Z(I+1),
     +                  ARRAY(I+1), MODE(I+1), SRDIST(I+1)
           HOLD = PC(I+1)
           IREC = IREC + 1
           GO TO 20
        END IF
C
        IF(PC(I) .NE. PC(I+1)) THEN
           PC(1) = PC(I+1)
           X(1) = X(I+1)
           Y(1) = Y(I+1)
           Z(1) = Z(I+1)
           ARRAY(1) = ARRAY(I+1)
           MODE(1) = MODE(I+1)
           SRDIST(1) = SRDIST(I+1)
           I = 1
        END IF
C
        GOTO 10
 20     I = I + 1
        GO TO 10
C
C
 30     WRITE(CRT,*)' THERE IS AN ERROR IN THE DATA FILE IN REC.',NREC
        WRITE(CRT,*)' ... OPERATION TERMINATING DUE TO ERROR. '
        STOP
 40     CONTINUE
C
C
        CLOSE(UNIT=1)
        CLOSE(UNIT=2)
C
 100    FORMAT(1X,I5,2X,F7.1,2X,2F9.1,2X,I2,2X,I2,2X,F8.2)
C
        RETURN
        END
C
C
C
C
C
C
```

```
      SUBROUTINE REC(CRT,KBD,TEMP4)
C     ***********************************************************
C     ***  Program to produce the final gating of Keyport   ***
C     ***  hydrophone array test data.                      ***
C     ***********************************************************
C
C
      INTEGER PC, ARRAY, PC1, PC2, A1, A2, ARRAY1, ARRAY2, CRT
      INTEGER PCA(10), ARRAYA(10), HOLD
      DIMENSION XA(10), YA(10), ZA(10), SRDISTA(10)
      CHARACTER   OUTFIL*13, ANS*1, TEMP4*9, TEMP1*9,RANGE*7
C
C
      RANGE='NANOOSE'
C
C
 99   WRITE(CRT,*)' What is the name you wish to give to ',
     +              ' your output file? '
      READ(CRT,'(A)') OUTFIL
C
      WRITE(CRT,*)' Enter the numbers of the arrays to be paired: '
      READ(KBD,*) A1, A2
C
C
      OPEN(1,FILE=TEMP4,STATUS='OLD')
      OPEN(2,FILE='TEMP1.DAT',STATUS='OLD')
C
C
 5    READ(1,*,END=8,ERR=8) PC, X, Y, Z, ARRAY, MODE, SRDIST
      IF((ARRAY .EQ. A1).OR.(ARRAY .EQ. A2)) THEN
         WRITE(2,100) PC, X, Y, Z, ARRAY,SRDIST
      END IF
      GOTO 5
C
 8    CONTINUE
      CLOSE (UNIT=1)
C
C     ...  Routine to pair data again:
      REWIND 2
      OPEN(4,FILE='TEMP2.DAT',STATUS='OLD')
C
      I = 1
      IFLAG = 0
      FIRST = 1
      HOLD = 0
      M = 0
C
 9    READ(2,100,END=40) PCA(I), XA(I), YA(I), ZA(I),
     +                   ARRAYA(I), SRDISTA(I)
        IF(FIRST .EQ. 1) THEN
          FIRST = 0
          HOLD = PCA(I)
        END IF

        IF(PCA(I) .EQ. HOLD) THEN
          I = I + 1
```

```
                  M = M + 1
                  GOTO 9
                ELSE
C
C                     ...In cases where there are three or more reports for
C                     ...   a given point count, segregate by comparing SRDIST
C                     ...   if there are more than 3 reports, discard all.
C
                    IF (M .EQ. 3) THEN
                      IF(ARRAYA(M) .EQ. ARRAYA(M-1)) THEN
                        IF (ARRAYA(M) .EQ. ARRAYA(M-2)) THEN
                          GOTO 50
                        ELSE
                          IF (ABS(SRDISTA(M-2)-SRDISTA(M)) .LT.
     +                        ABS(SRDISTA(M-2)-SRDISTA(M-1))) THEN
                            WRITE(4,100) PCA(M), XA(M), YA(M), ZA(M),
     +                                   ARRAYA(M), SRDISTA(M)
                            WRITE(4,100) PCA(M-2), XA(M-2), YA(M-2), ZA(M-2),
     +                                   ARRAYA(M-2), SRDISTA(M-2)
                            IFLAG = 1
                          ELSE
                            WRITE(4,100) PCA(M-1), XA(M-1), YA(M-1), ZA(M-1),
     +                                   ARRAYA(M-1), SRDISTA(M-1)
                            WRITE(4,100) PCA(M-2), XA(M-2), YA(M-2), ZA(M-2),
     +                                   ARRAYA(M-2), SRDISTA(M-2)
                            IFLAG = 1
                          END IF
                        END IF
                      ELSE
                          IF (ARRAYA(M) .EQ. ARRAYA(M-2)) THEN
                           IF (ABS(SRDISTA(M-1)-SRDISTA(M)) .LT.
     +                         ABS(SRDISTA(M-1)-SRDISTA(M-2))) THEN
                            WRITE(4,100) PCA(M), XA(M), YA(M), ZA(M),
     +                                   ARRAYA(M), SRDISTA(M)
                            WRITE(4,100) PCA(M-1), XA(M-1), YA(M-1), ZA(M-1),
     +                                   ARRAYA(M-1), SRDISTA(M-1)
                            IFLAG = 1
                          ELSE
                            WRITE(4,100) PCA(M-1), XA(M-1), YA(M-1), ZA(M-1),
     +                                   ARRAYA(M-1), SRDISTA(M-1)
                            WRITE(4,100) PCA(M-2), XA(M-2), YA(M-2), ZA(M-2),
     +                                   ARRAYA(M-2), SRDISTA(M-2)
                            IFLAG = 1
                          END IF
                         ELSE
                           IF (ABS(SRDISTA(M)-SRDISTA(M-1)) .LT.
     +                         ABS(SRDISTA(M)-SRDISTA(M-2))) THEN
                            WRITE(4,100) PCA(M), XA(M), YA(M), ZA(M),
     +                                   ARRAYA(M), SRDISTA(M)
                            WRITE(4,100) PCA(M-1), XA(M-1), YA(M-1), ZA(M-1),
     +                                   ARRAYA(M-1), SRDISTA(M-1)
                            IFLAG = 1
                          ELSE
                            WRITE(4,100) PCA(M), XA(M), YA(M), ZA(M),
     +                                   ARRAYA(M), SRDISTA(M)
                            WRITE(4,100) PCA(M-2), XA(M-2), YA(M-2), ZA(M-2),
     +                                 ARRAYA(M-2), SRDISTA(M-2)
                           IFLAG = 1
                         END IF
                       END IF
                 END IF
C
C
                 ELSE
                    IF ((M .EQ. 2) .AND. (ARRAYA(M) .NE. ARRAYA(M-1))) THEN
                      WRITE(4,100) PCA(M), XA(M), YA(M), ZA(M),ARRAYA(M),
     +                             SRDISTA(M)
                      WRITE(4,100) PCA(M-1), XA(M-1), YA(M-1), ZA(M-1),
     +                             ARRAYA(M-1), SRDISTA(M-1)

                    END IF
              END IF
 50              CONTINUE
C
C
 25              CONTINUE
C
                   IFLAG = 0
                   PCA(1) = PCA(I)
                   XA(1) = XA(I)
                   YA(1) = YA(I)
                   ZA(1) = ZA(I)
                   ARRAYA(1) = ARRAYA(I)
                   SRDISTA(1) = SRDISTA(I)
                   I = 2
                   M = 1
                   HOLD = PCA(1)
C
                   DO 45 L = 2,10
                   PCA(L) = 0
                   XA(L) = 0
                   YA(L) = 0
                   ZA(L) = 0
                   ARRAYA(L) = 0
                   SRDISTA(L) = 0
 45                CONTINUE
C
                 END IF
C
                 GOTO 9
 40         CONTINUE
C
C
        NREC = 0
C
        CLOSE (UNIT=4)
C
        OPEN(7,FILE='TEMP2.DAT',STATUS='OLD')
        OPEN(9,FILE=OUTFIL,STATUS='NEW')
C
        WRITE(9,300) RANGE, A1, A2
C       ... Read in array data in two record pairs:
 10     READ(7,100,END=60,ERR=70) PC1, X1, Y1, Z1, ARRAY1, SRDIST1
        READ(7,100,END=60,ERR=70) PC2, X2, Y2, Z2, ARRAY2, SRDIST2
C
C
        IF(ARRAY1 .EQ. A1 .AND. ARRAY2 .EQ. A2) THEN
C            ... If arrays are in specified order (e.g. 4,5):
             WRITE(9,200) PC1, X1, Y1, Z1, X2, Y2, Z2
C
        END IF
        IF(ARRAY1 .EQ. A2 .AND. ARRAY2 .EQ. A1) THEN
C            ... If arrays are in reverse order (eg. 5,4):
             WRITE(9,200) PC1, X2, Y2, Z2, X1, Y1, Z1
C
        END IF
C
C
C       ... Increment record counter:
        NREC = NREC + 1
C
C
        GOTO 10
C
C
 70     WRITE(CRT,*)' There is a bad record in the file.'
 60     CONTINUE
C
C
        CLOSE(UNIT=2)
        CLOSE(UNIT=7)
        CLOSE(UNIT=9)
C
C
 100    FORMAT(I5,3(2X,F8.1),2X,I2,2X,F8.2)
 200    FORMAT(2X,I5,1X,6(2X,F11.1))
 300    FORMAT(16X,A10,2X,I2,3X,I2)
C
        WRITE(CRT,*) ' Do you want to try another array pair? (Y/N)'
        READ(KBD,'(A)') ANS
        IF(ANS .EQ. 'Y' .OR.  ANS .EQ. 'y') GO TO 99
C
C
        RETURN
        END
```

APPENDIX B. PROGRAM TOE

```
      PROGRAM TOE
C***********************************************************************
C*                                                                     *
C* Programmer  : Sukru KORLU                                           *
C*                                                                     *
C* Purpose     : To calculate and eliminate timing  offset error by    *
C*               least squares in the keyport hydrophone data.         *
C*               Output of this program can be read by the program     *
C*               keymain3.                                             *
C*                                                                     *
C***********************************************************************
C
C
      DIMENSION DEPTH(53), VEL(53), L(53), G(53)
C
      INTEGER ARR1, ARR2, NL, PC, KBD, CRT
C
      REAL*8 X3, Y3, V0, V1, DEPTH, VEL, DL, L, G, LX1, LX2, LX3, LY1,
     +       LY2, LY3, P1, P2, A1, A2, GG, TTHETA, XTHETA, YTHETA,
     +       TVEL, XVEL, YVEL
C
      CHARACTER*12 FNAME1, FNAME2, FNAME3, FNAME4, FNAME5, FNAME6,
     +             FNAME7, FNAME8
C
      CHARACTER*1 ANS
C
      LOGICAL EXT, EXT3, EXT4, EXT5, EXT7
C
      PARAMETER(KBD=5 , CRT=6 , NL=53)
C
C
C Define input/output files...
C
      WRITE(CRT,*) ' Please enter the name of the range configuration',
     +' file...: '
      READ(KBD,'(A)') FNAME1
      INQUIRE(FILE=FNAME1,EXIST=EXT)
      IF(.NOT. EXT) THEN
      WRITE(CRT,*) ' The range config. file does not exist... '
      GOTO 99
      END IF
C
      WRITE(CRT,*) ' Please enter the name of the DV file...: '
      READ(KBD,'(A)') FNAME6
      INQUIRE(FILE=FNAME6,EXIST=EXT)
      IF(.NOT. EXT) THEN
      WRITE(CRT,*) ' The DV file does not exist... '
      GOTO 99
      END IF
C
      WRITE(CRT,*) ' Please enter the name you wish to give to your',
     +' output file...: '
      READ(KBD,'(A)') FNAME8
      INQUIRE(FILE=FNAME8,EXIST=EXT)
      IF(EXT) THEN
      WRITE(CRT,*) ' The output file already exists... '
      GOTO 99
      END IF
C
   10 WRITE(CRT,*) ' Please enter the name of the input file...: '
      READ(KBD,'(A)') FNAME2
      INQUIRE(FILE=FNAME2,EXIST=EXT)
      IF(.NOT. EXT) THEN
      WRITE(CRT,*) ' The input file does not exist... '
      GOTO 99
      END IF
C
      FNAME3 = 'LOCAL.REC'
      FNAME4 = 'TRANS1.REC'
      FNAME5 = 'TRANS2.REC'
      FNAME7 = 'THETA.REC'
      INQUIRE(FILE=FNAME3,EXIST=EXT3)
      INQUIRE(FILE=FNAME4,EXIST=EXT4)
      INQUIRE(FILE=FNAME5,EXIST=EXT5)
      INQUIRE(FILE=FNAME7,EXIST=EXT7)
      IF(EXT3 .OR. EXT4 .OR. EXT5 .OR. EXT7) THEN
      WRITE(CRT,*) ' Some of the intermediate files alredy exsist... '
      GOTO 99
      END IF
C
C
C Check input file get ok. from the user...
C
      OPEN(1,FILE=FNAME2,STATUS='OLD')
C
      READ(1,100) ARR1, ARR2
      WRITE(CRT,*) ' The arrays in the file are: ',ARR1,ARR2
      WRITE(CRT,*) ' Is this the file you want ? (Y/N) '
      READ(KBD,'(A)') ANS
C
      IF(ANS .NE. 'Y' .OR. ANS .NE. 'y') THEN
        WRITE(CRT,*) ' Redefine input data file... '
        CLOSE(UNIT=1)
        GOTO 10
      END IF
C
      CLOSE(UNIT=1)
C
C
C Begin to call subroutines...
C
      CALL LOCAL(FNAME1,FNAME2,FNAME3,ARR1,ARR2,X3,Y3,CRT)
C
```

```
      CALL TRANSFRM(FNAME3,FNAME4,FNAME5,CRT)
C
      CALL LSLINE(FNAME6,V0,V1,NL,CRT)
C
C
      OPEN(2,FILE=FNAME6,STATUS='OLD')
C
C Read depth velocity profile...
C
      WRITE(CRT,*) ' Setting layer end points and gradient values... '
      READ(2,*)
      READ(2,*)
C
      DL = 25.0D0
C
      DO 20 M = 1 , 52
      READ(2,200) DEPTH(M),VEL(M)
   20 CONTINUE
C
      CLOSE(UNIT=2)
C
      DEPTH(53) = DEPTH(52)+DL
      VEL(53) = 2.0D0*VEL(52)-VEL(51)
C
C Setting layer end points and gradient values...
C
      DO 30 M = 1 , 52
      L(M) = DEPTH(M)-(DL/2.0D0)
      G(M) = VEL(M+1)-VEL(M)
   30 CONTINUE
C
      GG = 0.0D0
      DO 40 M = 51 , 44 , -1
      GG = GG+(G(M+1)-G(M))
   40 CONTINUE
      GG = GG/8.0D0
      L(53) = L(52)+DL
      G(53) = G(52)+GG
C
      DO 50 M = 1 , 53
      G(M) = G(M)/DL
   50 CONTINUE
C
C Read position of the torpedo according to the left and the rigth array
C then call subroutine RTRACE for raytracing...
C
      OPEN(3,FILE=FNAME3,STATUS='OLD')
      OPEN(4,FILE=FNAME7,STATUS='NEW')
C
      WRITE(CRT,*) ' Reading the position of the torpedo... '
C
   60 READ(3,300,END=80,ERR=70) PC,LX1,LX2,LX3,LY1,LY2,LY3
C
      WRITE(CRT,*) ' Multilayer ray tracing... '
C
      P1 = DSQRT((LX1**2)+(LX2**2))
      P2 = DABS(X3)-LX3
      A1 = 0.0D0
      A2 = DABS(X3)
C
      CALL RTRACE(NL,L,VEL,DEPTH,A1,A2,P1,P2,DL,G,V0,V1,TTHETA,TVEL)
      XTHETA = TTHETA
      XVEL = TVEL
C
      P1 = DSQRT((LY1**2)+(LY2**2))
      P2 = DABS(Y3)-LY3
      A1 = 0.0D0
      A2 = DABS(Y3)
C
      CALL RTRACE(NL,L,VEL,DEPTH,A1,A2,P1,P2,DL,G,V0,V1,TTHETA,TVEL)
      YTHETA = TTHETA
      YVEL = TVEL
C
      WRITE(4,400) PC,XTHETA,XVEL,YTHETA,YVEL
      GOTO 60
   70 WRITE(CRT,*) 'There is an error in the local track file... '
      CLOSE(UNIT=3)
      CLOSE(UNIT=4)
      GOTO 99
   80 CLOSE(UNIT=3)
      CLOSE(UNIT=4)
C
C
C
      CALL CDELTA(FNAME2,FNAME4,FNAME5,FNAME7,FNAME8,CRT)
C
C
      WRITE(CRT,*) ' Operation complete. Program terminating... '
C
C
      STOP
C
   99 WRITE(CRT,*) ' OPERATION TERMINATING DUE TO THE ERROR !!! '
C
      STOP
C
C
  100 FORMAT(28X,I2,3X,I2)
  200 FORMAT(8X,F4.0,1X,F9.4)
  300 FORMAT(I5,1X,6(2X,F10.2))
  400 FORMAT(I5,3X,4(F15.9,2X))
C
C
C
      END
C
C
C
C
C
C
```

```
      SUBROUTINE LOCAL(NAME1,NAME2,NAME3,ARR1,ARR2,X3,Y3,CRT)
C***********************************************************************
C*                                                                     *
C* Programmer   : Sukru KORLU                                          *
C*                                                                     *
C* Purpose      : To calculate local track by using global track and   *
C*                array locations.                                     *
C*                                                                     *
C* Key variables                                                       *
C*     NAME1    : Name of the input file which carries the pair of     *
C*                global track.                                        *
C*     NAME2    : Name of the range configuration file.                *
C*     NAME3    : Name of the output file which carries the pair of    *
C*                local track.                                         *
C*     ARR1     : Left array number.                                   *
C*     ARR2     : right array number.                                  *
C*     PC       : Point count                                          *
C*     GX1-GX3  : Global track from left array.                        *
C*     GY1-GY3  : Global track from rigth array.                       *
C*     X-Y-Z    : Array location.                                      *
C*     ARRNUM   : Array number.                                        *
C*     X1-X3    : Left array location.                                 *
C*     Y1-Y3    : Rigth array location.                                *
C*     LX1-LX3  : Local track from the left array.                     *
C*     LY1-LY3  : Local track from the rigth array.                    *
C*     Q1,Q2    : Logical constants.                                   *
C*                                                                     *
C***********************************************************************
C
      INTEGER ARR1, ARR2, ARRNUM, Q, PC, CRT
C
      REAL*8 GX1, GX2, GX3, GY1, GY2, GY3, X, Y, Z, X1, X2, X3, Y1,
     +       Y2, Y3, LX1, LX2, LX3, LY1, LY2, LY3
C
      CHARACTER*12 NAME1, NAME2, NAME3
C
C Read array locations....
C
      OPEN(1,FILE=NAME1,STATUS='OLD')
C
      Q=0
      WRITE(CRT,*) ' Reading array locations... '
C
   10 READ(1,100,END=30,ERR=20)  X,Y,Z,ARRNUM
C
      IF(ARRNUM .EQ. ARR1) THEN
         X1 = X
         X2 = Y
         X3 = Z
         Q = Q+1
      END IF
C
      IF(ARRNUM .EQ. ARR2) THEN
         Y1 = X
         Y2 = Y
         Y3 = Z
         Q = Q+1
      END IF
C
      GOTO 10
   20 WRITE(CRT,*) ' There is an error in the range configuration',
     +             ' file... '
      CLOSE(UNIT=1)
      GOTO 99
   30 CLOSE(UNIT=1)
C
C Make sure that both array locations were found...
C
      IF(Q .NE. 2) THEN
         WRITE(CRT,*) ' At least one array location can not be found',
     +                ' in the range configuration file... '
         CLOSE(UNIT=1)
         GOTO 99
      END IF
C
C Calulate local track...
C
      OPEN(2,FILE=NAME2,STATUS='OLD')
      OPEN(3,FILE=NAME3,STATUS='NEW')
C
      WRITE(CRT,*) ' Reading global track writing local track...'
      READ(2,*)
C
   40 READ(2,200,END=60,ERR=50) PC,GX1,GX2,GX3,GY1,GY2,GY3
      LX1 = GX1-X1
      LX2 = GX2-X2
      LX3 = GX3-X3
      LY1 = GY1-Y1
      LY2 = GY2-Y2
      LY3 = GY3-Y3
      WRITE(3,300) PC,LX1,LX2,LX3,LY1,LY2,LY3
      GOTO 40
   50 WRITE(CRT,*) ' There is an error in the input file...'
      CLOSE(UNIT=2)
      CLOSE(UNIT=3)
      GOTO 99
   60 WRITE(CRT,*) ' Transformation done... '
C
      CLOSE(UNIT=2)
      CLOSE(UNIT=3)
C
      RETURN
C
C
   99 WRITE(CRT,*) 'OPERATION TERMINATING DUE TO THE ERROR !!! '
      STOP
C
  100 FORMAT(F10.4,3X,F11.3,6X,F11.3,6X,I2)
  200 FORMAT(2X,I5,1X,6(2X,F11.1))
  300 FORMAT(I5,1X,6(2X,F10.2))
C
```

```
      END
C
C
C
      SUBROUTINE TRANSFRM(NAME1,NAME2,NAME3,CRT)
C************************************************************************
C*                                                                      *
C* Programmer   : Sukru KORLU                                           *
C*                                                                      *
C* Purpose      : To make transform from cartesian coordinates to the   *
C*                azimuth,horizontal and vertical components.           *
C*                                                                      *
C* Key Variables                                                        *
C*      NAME1   : Name of the file that carries the pair of local track.*
C*      NAME2-3 : Names of the output files.                            *
C*      PC      : Point count.                                          *
C*      LX1-LX3 : Local track from the left array.                      *
C*      LY1-LY3 : Local track from the rigth array.                     *
C*      FIX     : Azimuth component of the left array track.            *
C*      FIY     : Azimuth component of the rigth array track.           *
C*      HORX    : Horizontal component of the left array track.         *
C*      HORY    : Horizontal component of the rigth array track.        *
C*      VERX    : Vertical component of the left array track.           *
C*      VERY    : Vertical component of the rigth array track.          *
C*                                                                      *
C************************************************************************
C
      INTEGER PC, CRT
C
      REAL*8 LX1, LX2, LX3, LY1, LY2, LY3, FIX, HORX, VERX, FIY,
     +       HORY, VERY, PI
C
      CHARACTER*12 NAME1, NAME2, NAME3
C
      PARAMETER(PI=3.141592654D0)
C
      OPEN(1,FILE=NAME1,STATUS='OLD')
      OPEN(2,FILE=NAME2,STATUS='NEW')
      OPEN(3,FILE=NAME3,STATUS='NEW')
C
      WRITE(CRT,*) ' Making tranform...'
C
   10 READ(1,100,END=30,ERR=20) PC,LX1,LX2,LX3,LY1,LY2,LY3
      FIX = DATAN(LX2/LX1)
      IF(LX1 .LT. 0) FIX=FIX+PI
      HORX = LX1/DCOS(FIX)
      VERX = LX3
      FIY = DATAN(LY2/LY1)
      IF(LY1 .LT. 0) FIY=FIY+PI
      HORY = LY1/DCOS(FIY)
      VERY = LY3
      WRITE(2,200) PC,FIX,HORX,VERX
      WRITE(3,200) PC,FIY,HORY,VERY
      GOTO 10
   20 WRITE(CRT,*) ' There is an error in the local track file...'
      CLOSE(UNIT=1)
      CLOSE(UNIT=2)
      CLOSE(UNIT=3)
      GOTO 99
   30 CLOSE(UNIT=1)
      CLOSE(UNIT=2)
      CLOSE(UNIT=3)
C
      RETURN
C
C
   99 WRITE(CRT,*) 'OPERATION TERMINATING DUE TO THE ERROR !!!'
      STOP
C
  100 FORMAT(I5,1X,6(2X,F10.2))
  200 FORMAT(I5,3X,3(F15.8,2X))
C
C
      END
C
C
C
      SUBROUTINE LSLINE(FNAME,V0,V1,NL,CRT)
C***********************************************************************
C*                                                                     *
C* Programmer  : Sukru KORLU                                           *
C*                                                                     *
C* Purpose     : To calculate V0 (speed of sound at the surface) and   *
C*               V1 (gradient) by least squares.                       *
C*                                                                     *
C* Key variables                                                       *
C*     FNAME   : Name of the file which carries depth velocity         *
C*               profile.                                              *
C*     X       : Depth.                                                *
C*     SX      : Sum of x.                                             *
C*     NL      : # of layers.                                          *
C*     XB      : Mean of x.                                            *
C*     SSX     : Sum of square of x.                                   *
C*     Y       : Speed of sound in the water.                          *
C*     SY      : Sum of y.                                             *
C*     YB      : Mean of y.                                            *
C*     SXY     : Sum of product of x and y.                            *
C*     V0      : Sound velocity at the surface.                        *
C*     V1      : Gradient.                                             *
C*                                                                     *
C***********************************************************************
C
      INTEGER NL, CRT
C
      REAL*8 V0, V1, X, SX, SSX, XB, Y, SY, YB, XY, SXY
C
      CHARACTER FNAME*12
C
      OPEN(1,FILE=FNAME,STATUS='OLD')
C
      WRITE(CRT,*) ' Calculating V0,V1 ... '
      READ(1,*)
      READ(1,*)
C
      SX = 0.0D0
      SSX = 0.0D0
      SY = 0.0D0
      SXY = 0.0D0
C
      DO 10 I = 1 , NL-1
      READ(1,100) X,Y
      SX = SX+X
      SSX = SSX+(X**2)
      SY = SY+Y
      SXY = SXY+X*Y
   10 CONTINUE
C
      XB = SX/(NL-1)
      YB = SY/(NL-1)
      V1 = (SXY-((NL-1)*XB*YB))/(SSX-((NL-1)*(XB**2)))
      V0 = YB-(V1*XB)
C
      CLOSE(UNIT=1)
C
      RETURN
C
  100 FORMAT(8X,F4.0,1X,F9.4)
C
      END
C
C
C
C
      SUBROUTINE RTRACE(NL,L,V,DEPTH,A1,A2,P1,P2,DL,G,V0,V1,TTHETA,TVEL)
C***********************************************************************
C*                                                                     *
C* Programmer     : Sukru KORLU                                        *
C*                                                                     *
C* Purpose        : To calculate elevation angle at the torpedo by     *
C*                  multilater raytracing for one point count.         *
C*                  Also to find speed of sound at the depth of        *
C*                  torpedo.                                           *
C*                                                                     *
C* Key variables                                                       *
C*      L         : An array which indicates layer end points.         *
C*      V         : An array which indicates speed of sound.           *
C*      G         : An array which indicates gradient values.          *
C*      THETA     : An array which indicates elevation angle in each   *
C*                  layer.                                             *
C*      RV        : Ray invariant.                                     *
C*      R         : Range of torpedo from the array.                   *
C*      A1,A2     : Array position.                                    *
C*      P1,P2     : Torpedo position.                                  *
C*      V0,V1     : Constants.                                         *
C*      C1,C2     : Center of the circle that we assumed sound travels *
C*                  on it's arc.                                       *
C*                                                                     *
C***********************************************************************
```

```
C
C
      DIMENSION LA(53), L(53), VE(53), V(53), DEPTH(53), G(53), GR(53),
     +          THETA(53), VVO(53), C2(53), T(53)
C
      INTEGER NL
C
      REAL*8 EP, LA, L, VE, V, A1, A2, P1, P2, G, VO, V1, GR, TVEL, C1,
     +      C2, CC2, THETA, ATHETA, TTHETA, OTHETA, VVO, T, RV, R, DL,
     +      DEPTH
C
C Find the layers that torpedo or array in it, redefine the end points
C of those layers, define tolerance...
      EP = 0.1D-6
      I = 0
      J = 0
C
      DO 10 K = 1 , NL
      LA(K) = L(K)
      VE(K) = V(K)
      GR(K) = G(K)
      IF(LA(K) .LE. A2) J=J+1
      IF(LA(K) .LE. P2) I=I+1
   10 CONTINUE
C
      IF(DEPTH(I) .LE. P2) THEN
         VE(I)=VE(I)+GR(I)*(P2-DEPTH(I))
      ELSE
         VE(I)=VE(I)-GR(I-1)*(DEPTH(I)-P2)
      END IF
C
      IF(DEPTH(J) .LE. P2) THEN
         VE(J)=VE(J)+GR(J)*(A2-DEPTH(J))
      ELSE
         VE(J)=VE(J)-GR(J-1)*(DEPTH(J)-A2)
      END IF
C
      LA(I) = P2
      LA(J) = A2
      TVEL = VE(I)
C
C Calculate an initial estimate for the elevation angle using single
C layer approximation...
      CC2 = -VO/V1
      C1 = (0.5D0*(P1+A1))+(0.5D0*((LA(I)-LA(J))*(((LA(I)+LA(J)-2.0D0*
     +CC2))/(P1-A1))))
      ATHETA = DATAN((A1-C1)/(A2-CC2))
C
C Using this initial estimate of the elevation angle with ray invariant
C to raytrace back through all the layers...
   20 RV = DCOS(ATHETA)/VE(J)
      DO 30 K = I , J
      THETA(K) = DACOS(RV*VE(K))
      T(K) = DTAN(THETA(K))
      VVO(K) = VE(K)-LA(K)*GR(K)
      C2(K) = -VVO(K)/GR(K)
   30 CONTINUE
C
C Using the angle just calculated iterate backwards through the layers
C from array to torpedo to get the horizontal range...
      R = 0.0D0
      DO 40 K = J ,(I+1), -1
      C1 = R-T(K)*(LA(K)-C2(K-1))
      R = C1+T(K-1)*(LA(K-1)-C2(K-1))
   40 CONTINUE
C
C Test if the value for the range within tolerance if not,redefine ATHETA
C the initial angle and raytrace again...
      IF((DABS(R-P1)) .LE. EP) GOTO 50
      OTHETA = ATHETA
      ATHETA = DATAN(DTAN(OTHETA)*(R/P1))
      GOTO 20
C
   50 TTHETA = THETA(I)
C
      RETURN
C
      END
C
C
C
      SUBROUTINE CDELTA(NAME1,NAME2,NAME3,NAME4,NAME5,CRT)
C*********************************************************************
C*                                                                   *
C* Programmer    : Sukru KORLU                                       *
C*                                                                   *
C* Purpose       : To calculate timing offset error(delta) by least  *
C*                 squares.                                          *
C*                                                                   *
C* Key variables                                                     *
C*     NAME1     : Name of the file which carries global track from  *
C*                 pair of arrays.                                   *
C*     NAME2     : Name of the file which carries transformed track  *
C*                 from left array.                                  *
C*     NAME3     : Name of the file which carries transformed track  *
C*                 from rigth array.                                 *
C*     NAME4     : Name of the file which carries elevation angle and *
C*                 sound velocity records.                           *
C*     NAME5     : Output file which carries corrected track.        *
C*     SSQ       : Sum of squares.                                   *
C*     SPR       : Sum of products.                                  *
C*     X         : Track from the left array.                        *
C*     Y         : Track from the rigth array.                       *
C*     FIX-FIY   : Azimuth angles.                                   *
C*     XVEL-YVEL : Speed of sound at the depth of torpedo.           *
C*     X/YTHETA  : Elevation angles.                                 *
C*     DELTA     : Timing offset error.                              *
C*                                                                   *
C*********************************************************************
C
C
```

```
      DIMENSION X(3), Y(3), XY(3), A(100,3), B(100,3), AB(3)
C
      INTEGER PC, CRT
C
      REAL*8 X, Y, FIX, FIY, XTHETA, YTHETA, XVEL, YVEL, A, B, AB,
     +       XY, SSQ, SPR, DELTA
C
      CHARACTER*12 NAME1, NAME2, NAME3, NAME4, NAME5
C
      CHARACTER*35 AUTL
C
C
      WRITE(CRT,*) ' Calculating delta... '
C
      OPEN(1,FILE=NAME1,STATUS='OLD')
      OPEN(2,FILE=NAME2,STATUS='OLD')
      OPEN(3,FILE=NAME3,STATUS='OLD')
      OPEN(4,FILE=NAME4,STATUS='OLD')
C
C
      I = 0
      SSQ = 0.0D0
      SPR = 0.0D0
C
      READ(1,*)
C
   10 READ(1,100,END=30,ERR=20) X(1),X(2),X(3),Y(1),Y(2),Y(3)
      I = I+1
      READ(2,200) FIX
      READ(3,200) FIY
      READ(4,300) XTHETA,XVEL,YTHETA,YVEL
      XY(1) = X(1)-Y(1)
      XY(2) = X(2)-Y(2)
      XY(3) = X(3)-Y(3)
      A(I,1) = XVEL*DCOS(XTHETA)*DCOS(FIX)
      A(I,2) = XVEL*DCOS(XTHETA)*DSIN(FIX)
      A(I,3) = XVEL*DSIN(XTHETA)
      B(I,1) = YVEL*DCOS(YTHETA)*DCOS(FIY)
      B(I,2) = YVEL*DCOS(YTHETA)*DSIN(FIY)
      B(I,3) = YVEL*DSIN(YTHETA)
      AB(1) = A(I,1)-B(I,1)
      AB(2) = A(I,2)-B(I,2)
      AB(3) = A(I,3)-B(I,3)
      SSQ = SSQ+(AB(1)**2)+(AB(2)**2)+(AB(3)**2)
      SPR = SPR+(XY(1)*AB(1))+(XY(2)*AB(2))+(XY(3)*AB(3))
      GOTO 10
   20 WRITE(CRT,*) ' There is an error in the input file... '
      CLOSE (UNIT=1)
      CLOSE (UNIT=2)
      CLOSE (UNIT=3)
      CLOSE (UNIT=4)
      GOTO 99
   30 CLOSE(UNIT=2)
      CLOSE(UNIT=3)
      CLOSE(UNIT=4)
C
```

```
      DELTA = SPR/SSQ
      WRITE(CRT,*) ' TIMING ERROR = ',DELTA
C
      REWIND 1
C
      OPEN(7,FILE=NAME5,STATUS='NEW')
C
      WRITE(CRT,*) ' Writing output file... '
C
      READ(1,400) AUTL
      WRITE(7,400) AUTL,DELTA
C
      J = 0
   40 READ(1,500,END=60,ERR=50) PC,X(1),X(2),X(3),Y(1),Y(2),Y(3)
      J = J+1
      X(1) = X(1)-DELTA*A(J,1)
      X(2) = X(2)-DELTA*A(J,2)
      X(3) = X(3)-DELTA*A(J,3)
      Y(1) = Y(1)-DELTA*B(J,1)
      Y(2) = Y(2)-DELTA*B(J,2)
      Y(3) = Y(3)-DELTA*B(J,3)
      WRITE(7,500) PC,X(1),X(2),X(3),Y(1),Y(2),Y(3)
      GOTO 40
   50 WRITE(CRT,*) ' There is an error in the input file... '
      CLOSE (UNIT=1)
      CLOSE (UNIT=7)
      GOTO 99
   60 CLOSE(UNIT=1)
      CLOSE(UNIT=7)
C
      RETURN
C
   99 WRITE(CRT,*) ' OPERATION TERMINATING DUE TO THE ERROR !!! '
      STOP
C
  100 FORMAT(8X,6(2X,F11.1))
  200 FORMAT(8X,F15.8)
  300 FORMAT(8X,4(F15.9,2X))
  400 FORMAT(A35,5X,F8.6)
  500 FORMAT(2X,I5,1X,6(2X,F11.1))
C
      END
```

APPENDIX C. APL CODE TO CALCULATE ARRAY DISPLACEMENT AND ORIENTATION CORRECTIONS

```
      ∇ V←LOC CORR SET
[1]    EPSA←0.01
[2]    X←SET[; 2 3 4]
[3]    Y←SET[; 5 6 7]
[4]    XB← 3 1 ρ((+/X)÷N←1↑ρX)
[5]    YB← 3 1 ρ((+/Y)÷N)
[6]    X̲←X-(ρX)ρXB
[7]    Y̲←Y-(ρY)ρYB
[8]    CXX←((⍉X)+.×X)÷N-1
[9]    CYX←((⍉Y)+.×X)÷N-1
[10]   CYY←((⍉Y)+.×Y)÷N-1
[11]   I←B← 3 3 ρ 1 0 0 0
[12]   D←YB-XB
[13]   D[3;1]←0
[14]   A←D-(B-I)+.×LOC
[15]   IQO←,(+/((Y̲-X̲+.×⍉B)*2))÷N
[16]   IQL←,(YB-A+B+.×XB)*2
[17]   IQ←+/IQO+IQL
[18]  L:Z←CYX+((YB-A)+.×⍉XB)
[19]   BB←B BMAX1 Z
[20]   B←BB
[21]   DD←YB-(LOC+B+.×(XB-LOC))
[22]   DD[3;1]←0
[23]   RA←(+/((A-AA)*2))*0.5
[24]   D←DD
[25]   →L×⍳EPSA<|RA
```

```
[26]  A←D-(B-I)+.×LOC
[27]  B1←3⍴0
[28]  LQO←,(+⌿((Y-X+.×⍉B)*2))÷N
[29]  LQL←,(YB-A+B+.×XB)*2
[30]  B1[2]←¯1○B[3;1]
[31]  B1[1]←¯1○(B[3;2]÷(2○B1[2]))
[32]  B1[3]←¯1○(B[2;1]÷(2○B1[2]))
[33]  A←,A
[34]  LQ←+/LQO+LQL
[35]  V←,D,B1
      ∇
```

```
      ∇ BB←B BMAX D
[1]   EPSB←1E¯6
[2]   W←W0←+/+/D×⍉B
[3]  L:WW←W
[4]   C←D+.×⍉B
[5]   BB←TWODIM C
[6]   BB←BB+.×B
[7]   W←+/+/D×⍉BB
[8]   R←W-WW
[9]   →L×⍳EPSB<|R
      ∇
```

```
        ∇ BB←TWODIM D;S;C;SS;B1;E;B2;D;F;B3
[1]     S←D[3;2]-D[2;3]
[2]     C←D[2;2]+D[3;3]
[3]     SS←((S×S)+C×C)*0.5
[4]     S←S÷SS
[5]     C←C÷SS
[6]     B1← 3 3 ρ 1 0 0 0 ,C,S,0,(-S),C
[7]     E←D+.×B1
[8]     S←E[3;1]-E[1;3]
[9]     C←E[3;3]+E[1;1]
[10]    SS←((S×S)+C×C)*0.5
[11]    S←S÷SS
[12]    C←C÷SS
[13]    B2← 3 3 ρC,0,S,0,1,0,(-S),0,C
[14]    F←E+.×B2
[15]    S←F[2;1]-F[1;2]
[16]    C←F[1;1]+F[2;2]
[17]    SS←((S×S)+C×C)*0.5
[18]    S←S÷SS
[19]    C←C÷SS
[20]    B3← 3 3 ρC,S,0,(-S),C, 0 0 0 1
[21]    BB←⍉B1+.×B2+.×B3
        ∇
```

APPENDIX D. APL CODE TO CALCULATE NONNEGATIVE COVARIANCE MATRICES

```
     - IND REGRESS DATA
      IND1←+/IND
      Y̲←(+/DATA)÷(1↑ρDATA)
      Y←((ρIND1),6)ρ0
      MSE←MSB← 6 6 ρ0
      D←0
      J←1
    L1:T←D+R←IND1[J]
      Y[J;]←(+/((D,0)↓((T,6)↑DATA)))÷R
      MSB←MSB+((Y[J;]-Y̲)∘.×(Y[J;]-Y̲))
      I←D+1
    L2:Y1←DATA[I;]-Y[J;]
      MSE←MSE+(Y1∘.×Y1)
      I←I+1
      →(I≤T)/L2
      D←T
      J←J+1
      →(J≤ρIND1)/L1
      MSB←MSB÷DFB←(ρIND1)-1
      MSE←MSE÷DFE←(+/IND1)-(((¯1↑ρIND)-1)+((1↑ρIND)-1))
      C←((+/IND1)-((+/(+/(IND*2)))÷(+/IND1)))÷((ρIND1)-1)
      EIE1←((,EIE← 1 6 ↑EE←EIGENR MSE)+1E¯20)*¯0.5
-22-  DE- 6 6 -EIE1-1-,0,0,0,0,0,0,EIE1-2-,0,0,0,0,0,0,EIE1-3-,0,0,0,
      0,0,0,EIE1-4-,0,0,0,0,0,0,EIE1-5-,0,0,0,0,0,0,EIE1-6-
      QE← 1 0 ↓EE
      LE←(QE+.×DE)+.×(⍉QE)
      L←(⍉LE)+.×(MSB+.×LE)
      EIL←,(1 6 ↑EL←EIGENR L)
```

```
[26]  EIL←,(1 6 ↑EL←EIGENR L)
[27]  K←+/A←EIL≥1
[28]  QL← 1 0 ↓EL
[29]  P←(⌹(⍉LE))+.×QL
[30]  EIL1←A/EIL
[31]  →(K>0)/L3
[32]  SIGMAB← 6 6 ρ0
[33]  →L4
[34] L3:PK←A/P
[35]  LAMDA←IDENT←(K,K)ρ1,B←Kρ0
[36]  X←1
[37] L5:LAMDA[X;]←IDENT[X;]×EIL1[X]
[38]  X←X+1
[39]  →(X≤K)/L5
[40]  SIGMAB←(PK+.×((LAMDA-IDENT)+.×(⍉PK)))÷C
[41] L4:SIGMAE←((DFB×(MSB-(C×SIGMAB)))+(DFE×MSE))÷(DFB+DFE)
      ∇
```

APPENDIX E. APL CODE TO ESTIMATE STABLE CORRECTIONS

```
      ∇ ALPHA←EST
-1-  ⍝ Y,SIGMAB,SIGMAE ARE GLOBAL VECTORS
[2]   X1← 7 2 ⍴ 1 0
[3]   X2← 2 2 ⍴ 0 1
[4]   X3← 1 2 ⍴ 1 0
[5]   J1← 7 7 ⍴1
[6]   J2← 2 2 ⍴1
[7]   J3← 1 1 ⍴1
[8]   A←PENROSE SIGMAE
[9]   D1←(PENROSE(SIGMAE+(7×SIGMAB)))+.×SIGMAB+.×A
[10]  D2←(PENROSE(SIGMAE+(2×SIGMAB)))+.×SIGMAB+.×A
[11]  D3←(PENROSE(SIGMAE+SIGMAB))+.×SIGMAB+.×A
[12]  Z← 6 6 ⍴0
[13]  X1A←⍉((⍉A),Z)
[14]  X1A←X1A,X1A,X1A,X1A,X1A,X1A,X1A
[15]  X2A←⍉(Z,(⍉A))
[16]  X2A←X2A,X2A
[17]  X3A←⍉((⍉A),Z)
[18]  XJ1←(⍉X1)+.×J1
[19]  XJ2←(⍉X2)+.×J2
[20]  XJ3←(⍉X3)+.×J3
[21]  XJD1←⍉((⍉D1),Z)
[22]  XJD1←XJD1,XJD1,XJD1,XJD1,XJD1,XJD1,XJD1
[23]  XJD2←⍉(Z,(⍉D2))
[24]  XJD2←XJD2,XJD2
[25]  XJD3←⍉((⍉D3),Z)
[26]  T2←(X1A-XJD1),(X2A-XJD2),(X3A-XJD3)
[27]  XX1←(⍉X1)+.×X1
[28]  XX2←(⍉X2)+.×X2
```

```
[29]  XX3←(⍉X3)+.×X3
[30]  XX1A←(⍉((⍉(7×A)),Z)),(⍉(Z,Z))
[31]  XX2A←(⍉(Z,Z)),(⍉(Z,(⍉(2×A))))
[32]  XX3A←(⍉((⍉A),Z)),(⍉(Z,Z))
[33]  XJX1←(⍉X1)+.×J1+.×X1
[34]  XJX2←(⍉X2)+.×J2+.×X2
[35]  XJX3←(⍉X3)+.×J3+.×X3
[36]  XJXD1←(⍉((⍉(49×D1)),Z)),(⍉(Z,Z))
[37]  XJXD2←(⍉(Z,Z)),(⍉(Z,(⍉(4×D2))))
[38]  XJXD3←(⍉((⍉D3),Z)),(⍉(Z,Z))
[39]  T1←(3×(XX1A+XX2A+XX3A))-(XJXD1+XJXD2+XJXD3)
[40]  T11←T1[⍳6;⍳6]
[41]  T12←T1[6+⍳6;6+⍳6]
[42]  T11[1;3]←T12[1;3]←0
[43]  IT11←PENROSE T11
[44]  IT12←PENROSE T12
[45]  ALPHA←(IT1←(⍉((⍉IT11),Z)),(⍉(Z,(⍉IT12))))+.×T2+.×Y
      ∇
```

LIST OF REFERENCES

1. Kinsler L. E., Frey A. R., Coppins A. B., and Sanders J. V., *Fundamentals of Acoustics*, Wiley Series, New York, 1982.

2. Read R. R., *Program for the Simultaneous Estimation of Displacement and Orientation Corrections for Several Short Base Line Arrays*, Naval Postgraduate School, Monterey CA., 1985.

3. Read R. R., "Remote Monitoring of the Calibration of a System of Tracking Arrays", *IEEE Journal of Oceanic Engineering*, Vol. OE-12, No. 1, January, 1987.

4. Main C. D., "Alternative Models for Calculation of Elevation Angles and Ray Transit Times for Ray Tracing of Hydrophonic Tracking Data", *Masters Thesis*, Naval Postgraduate School, Monterey CA., September, 1984.

5. Kroshl W. M., "Methodologies for Resolving Anomalous Position Information in Torpedo Range Tracking Using Simulation", *Masters Thesis*, Naval Postgraduate School, Monterey CA., March, 1988.

6. Searle S. R., *Linear Models*, Wiley Series in Probability and Mathemathical Statistics, New York, 1971.

7. Dahm P. F., Melton B. E. and Fuller W. A., "Generalized Least Squares Estimation of a Genotypic Covariance", *Biometrics*, Vol. 39, 1983

8. Amemiya Y., "What Should Be Done When an Estimated Between-Group Covariance Matrix Is not Nonnegative Definite", *The American Statistician*, Vol. 39, No. 2, May, 1985.

9. Graybill F. *Introduction to Matrices with Applications to Statistics*, Wadsworth, Belmont CA., 1969

10. Harville D. A., "Maximum Likelihood Approaches to Variance Component Estimation and to Related Problems", *Journal of American Statistical Association*, Vol. 72.1, 1977.

11. Searle S. R., "Another Look at Hendersons Methods of Estimating Variance Components", *Biometrics*, Vol. 24, 1968.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Deniz Kuvvetleri Komutanligi Personel Daire Baskanligi Bakanliklar-Ankara / TURKEY	1
4.	Deniz Harp Okulu Komutanligi Kutuphanesi Tuzla - Istanbul / TURKEY	1
5.	Professor R. R. Read, Code 55Re Operations Research Department Naval Postgraduate School Monterey, CA 93943	2
6.	Professor H. Larson, Code 55La Operations Research Department Naval Postgraduate School Monterey, CA 93943	1
7.	Sukru Korlu 21/B Blok Daire 3 Yenilevent - Istanbul / TURKEY	2